APPROXIMATE DYNAMIC PROGRAMMING APPROACH FOR SEQUENTIAL CHANGE DIAGNOSIS PROBLEM

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Elif Akbulut

August, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Savaş Dayanık(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Nesim K. Erkip

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Sinan Gezici

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

APPROXIMATE DYNAMIC PROGRAMMING APPROACH FOR SEQUENTIAL CHANGE DIAGNOSIS PROBLEM

Elif Akbulut
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Dr. Savaş Dayanık
August, 2013

We study sequential change diagnosis problem which is the combination of change diagnosis and multi-hypothesis testing problem. One observes a sequence of independent and identically distributed random variables. At a sudden disorder time, the probability distribution of the random variables change. The disorder time and its cause are unavailable to the observer. The problem is to detect this abrupt change in the distribution of the random process as quickly as possible and identify its cause as accurately as possible. Dayanık et al. [Dayanık, Goulding and Poor, Bayesian sequential change diagnosis, Mathematics of Operations Research, vol. 45, pp. 475-496, 2008] reduce the problem to a Markov optimal stopping problem and provide an optimal sequential decision strategy. However, only a small subset of the problems is computationally feasible due to curse of dimensionality. The subject of this thesis is to search for the means to overcome the curse of dimensionality. To this end, we propose several approximate dynamic programming algorithms to solve large change diagnosis problems. On several numerical examples, we compare their performance against the performance of optimal dynamic programming solution.

Keywords: Sequential change diagnosis, approximate dynamic programming, function approximation.

ÖZET

ARDIŞIK DEĞİŞİM VE TANI PROBLEMİ İÇİN YAKLAŞIK DİNAMİK PROGRAMLAMA YAKLAŞIMI

Elif Akbulut
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Savaş Dayanık
Ağustos, 2013

Bu araştırmada, değişim tanı ve çoklu denence sınamı problemlerinin birleşimi olan ardışık değişim tanı problemi çalışılmıştır. Bir gözlemci, bağımsız özdeşçe dağılmış rassal değişken dizisini gözlemler. Ani bir bozulma zamanında, rassal değişkenin olasılık dağılımı değişir. Bu değişimin zamanı ve nedeni gözlemci tarafından bilinmemektedir. Problem, bozulma zamanını ve bozulmanın nedenini mümkün olduğunca kısa zamanda ve doğru olarak bulmaktır. Dayanık ve ark. [Dayanık, Goulding and Poor, Bayesian sequential change diagnosis, Mathematics of Operations Research, vol. 45, pp. 475-496, 2008] problemi Markof eniyi durma problemine indirgedi ve eniyi ardışık karar stratejisini sundu. Ancak problem boyutu büyüdükçe problemler bu yolla çözülememektedir. Bu tezin amacı yaklaşık dinamik izlenceleme algoritmaları ile büyük boyutlu problemleri çözülebilir kılmaktır. Bir çok sayısal örnekte, yaklaşık eniyi izlenceleme algoritmalarının başarımı ile eniyi dinamik izlenceleme başarımı karşılaştırılmıştır.

Anahtar sözcükler: ardışık değişim tanı problemi, yaklaşık dinamik izlenceleme, işlev yaklaşıklama.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor Professor Savaş Dayanık for everything I learned from him. He has been always supportive and extremely patient throughout the course of this research. It was a valuable and great experience to work with him.

I thank TÜBİTAK for providing me financial support during my studies through the Grant 110M610.

Finally I would like to thank my husband, my best friend, Muhammed Akbulut for his continuous support. I am also truly grateful to my parents Sevim Keten and Mehmet Keten and my brother Emre Keten for their countless sacrifices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We study sequential change diagnosis problem by adopting approximate dynamic programming approach. Sequential change diagnosis problem is a combination of classical change diagnosis problem and multi-hypothesis testing. Change diagnosis problem consists of detection of the time of a sudden and unobservable change in the distribution of a random process. Multi-hypothesis testing focuses on identifying the change distribution. The aim of sequential change diagnosis problem is to both detect and identify the change in the distribution as quick as possible. The problem has a wide range of applications including healthcare, finance, quality control, fault detection and national defense. For example, Dayanık, Powell, and Yamazaki [1] study sequential change diagnosis problem in public health. They aim to detect a change in the distribution of sales of cough medicine and sales of tissues and identify whether the cause of the change is Anthrax or a common cold. It is crucial to detect and identify the change immediately if it is due to Anthrax since Anthrax is a fatal and highly infectious disease. A false alarm cost is incurred if one claims that the change is due to Anthrax and it is not. Observing more information gives the opportunity to make an accurate decision, however, it raises the risk that the disease enters the population. Hence, the problem includes trade-off between detection and identification delay cost and false alarm cost. Dayanık, Goulding and Poor [2] study the problem in a Bayesian framework. They show that sequential change diagnosis problem can

be turned into a Markov optimal stopping problem. By using this fact, they use tools of optimal stopping theory and submit the optimality equation for the problem. The dynamic programming (DP) formulation allows us to find the optimal solution. However, DP formulation works only in small size problems due to curse of dimensionality [3].

In this thesis, we study the sequential change diagnosis problem in a Bayesian set-up where the change time is assumed to be random. We use dynamic programming equation in [2] and employ approximate dynamic programming approach to avoid curse of dimensionality. Briefly, in approximate dynamic programming, we step forward in time by using a sample realization and use a value function approximation instead of exact value of the value function. In our study, we use basis functions for value function approximation. The set of basis functions we adopt is radial-basis penalized splines. Together with radial-basis penalized splines, linear mixed models are used to approximate value function. In dynamic programming, when we step backward in time, we always take the best decision. However, in approximate dynamic programming algorithm, we sometimes take a worse decision to explore unvisited states. This concept is referred as exploration vs. exploitation problem. We used a simple heuristic to decide when to take the best decision and when to explore unvisited states. In general, this thesis represents an approach to solve large size sequential change diagnosis problems in which DP formulation becomes intractable.

The remainder of the thesis is organized as follows. Chapter 2 reviews the related literature on sequential change diagnosis problem. In Chapter 3, we precisely formulate the problem and show that it can be treated as an optimal stopping problem. We also present dynamic programming optimality equation for the problem. Some background about classical dynamic programming and approximate dynamic programming is given in Chapter 4. Chapter 5 presents our solution methodology. We present both dynamic programming and approximate dynamic programming models of sequential change diagnosis problem. Experimental results are presented in Chapter 6. Chapter 7 concludes the thesis.

Chapter 2

Literature Review

Foundations of sequential change detection problem are discussed in Lorden [4] and Shiryayev [5]. Early problems of sequential hypothesis testing are studied by Wald and Wolfowitz [6] and Arrow, Blackwell, and Girshick [7]. Both change detection and sequential hypothesis testing problems have been studied extensively. Background information on sequential change detection and the design and investigation of change detection algorithms can be found in Basseville and Nikiforov [8] and Poor and Hadjiliadis [9]. On the other hand, the literature about sequential change diagnosis problem which is the combination of change detection problem and multi-hypothesis testing is sparse. The foundational studies of sequential change diagnosis problem are presented by Nikiforov [10] and Lai [11]. They study the problem in a non-Bayesian framework. Dayanık, Goulding, and Poor [2] study the problem in a Bayesian framework and provide non-asymptotic results. They show that the problem can be turned into a Markov optimal stopping problem. They come up with an optimality equation for the value function of the problem via tools of optimal stopping theory. They present the properties of optimal sequential decision strategy. They also provide illustrative examples showing geometric properties of optimality results. In [2], it is assumed that disorder distribution is geometric, disorder time is statistically independent on its cause and only a single change can be detected. This assumptions may not hold in some applications. Dayanık and Goulding [12] provide a more comprehensive

formulation. Their model allows a general disorder distribution. There are also some applications in which more than one changes occur. That is, the distribution of observations may change several times before we raise an alarm. The formulation of Dayanık and Goulding allows for multiple disorder times. Their model also removes the assumption that disorder time is statistically independent from its cause. A case in point is that if a change occurs within a specific period of time then the change type is most likely a specific one. Their formulation captures above relaxations both individually and concurrently. In this paper, Dayanık and Goulding study the problem via a hidden Markov model. They make a Bayesian posterior analysis and come up with an optimal solution for the problem. By using the optimality results in [2] and [12] optimal decision for the problem can be found via dynamic programming. However, the state space exponentially increases in the number of change types. Hence, finding an optimal strategy for the problems with large state space is computationally impossible by using tools of dynamic programming. Dayanık et al. [13] come up with two strategies that are computationally tractable and asymptotically optimal. First strategy is asymptotically optimal with respect to Bayes risk formulation. In Bayes risk formulation, the aim is to minimize a Bayes risk function which is the sum of the expected delay cost, expected false alarm cost and expected false identification cost. The second sequential decision strategy is asymptotically optimal in fixed-error formulation, which minimizes the expected delay cost subject to some upper bound on the expected false alarm.

The literature about dynamic programming and approximate dynamic programming is also related with our work. Dynamic programming [14] offers a general approach to formulate and solve sequential decision making problems. We refer reader to Bertsekas and Tsitsiklis [15], Sutton and Barto [16], and Powell [3] for a comprehensive reference.

Chapter 3

Sequential Change Diagnosis Problem

Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ hosts a stochastic process $X = (X_n)_{n\geq 1}$ which takes values in some measurable space $(E, \mathcal{E})$. Let $\theta : \Omega \mapsto \{0, 1, ...\}$ and $\mu : \Omega \mapsto \mathcal{M} \triangleq \{0, 1, ...\}$ be some random variables belonging to the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. An observer observes the process $X = (X_n)_{n\geq 1}$. A sudden change occurs at a disorder time θ. Before θ, $X_1, ..., X_{\theta-1}$ are independent and identically distributed with a common distribution $\mathbb{P}_0$. The initial distribution function $\mathbb{P}_0$ is known. After the disorder time θ, the process $X = (X_n)_{n\geq 1}$ shifts to one of the alternative change distributions $\mu = i$ where $i \in \{0, 1, ..., M\}$ and $X_\theta, X_{\theta+1}, ...$ are independent and identically distributed with a common distribution function $\mathbb{P}_\mu$. Suppose that for every $t \geq 1$, $i \in \mathcal{M}$, $n \geq 1$, and $(E_k)_{k=1}^n \subseteq \mathcal{E}$,

$$\begin{aligned}\mathbb{P}\{\theta = t, \mu = i, X_1 \in E_1, ..., X_n \in E_n\} \\ = (1 - p_0)(1 - p)^{t-1} p\nu_i \prod_{1\leq k\leq (t-1)\wedge n} \mathbb{P}_0(E_k) \prod_{t\vee 1\leq l\leq n} \mathbb{P}_i(E_l).\end{aligned} \tag{3.1}$$

We denote the probability that the change type is i as ν_i where $i \in \{0, 1, ..., M\}$

$$\nu_1 + \nu_1 + ... + \nu_M = 1. \tag{3.2}$$

The set of alternative change distributions (change types) consists of M different probability distributions and the random variable μ is the change index representing the change type. θ has a zero modified geometric distribution with p_0 and p.

$$\theta = \begin{cases} 0 & w.p \qquad\qquad\qquad\qquad p_0, \\ t & w.p \quad (1-p_0).(1-p)^{t-1}p \text{ for every } t = 1, 2, \dots \end{cases}$$

The observations are available to the observer sequentially. Each time an observation arrives, the observer takes the decision of raising an alarm or continuing to observe the system. When he raises an alarm he determines which type of a change occurred. Let $\mathbb{F} = (\mathcal{F})_{n\leq 0}$ denotes the natural filtration of the observation process X. $\mathcal{F}_0 = \{\emptyset, \Omega\}$ and $\mathcal{F}_n = \sigma(X_1, ..., X_n), n \geq 1$. $\delta = (\tau, d)$ is a sequential decision strategy which consists of a stopping rule τ of $\mathbb{F}$ and an isolation decision $d : \Omega \mapsto \mathcal{M}$ measurable with respect to the observation history $\mathcal{F}_\tau = \sigma(X_{n\wedge\tau}; n \geq 1)$. Namely, τ is the time when an alarm is raised and the observer claims that $\theta \leq \tau$.

Let Δ be the set of all possible sequential change strategies which is defined as follows.

$$\Delta \triangleq \{(\tau, d) | \tau \in \mathbb{F}, \text{and } d \in \mathcal{F}_\tau \text{ is an } \mathcal{M} - \text{valued random variable}\}.$$

The aim is to find the decision strategy $\delta = (\tau, d) \in \Delta$ which minimizes total expected cost. The cost is incurred by detection delay, false alarm and false isolation. We introduce these costs as follows.

Let c be the cost for each observation after the disorder time θ. a_{0j} denotes the cost associated with the decision $\delta = (\tau, d = j)$ when $\tau < \theta$, i.e., a_{0j} is the cost of a false alarm. Let a_{ij} be the cost associated with our decision $\delta = (\tau, d = j)$ when $\tau \geq \theta$ and $\mu = i$, i.e., false isolation cost. Since accurate isolation does not incur a cost, $a_{ii} = 0$ for every $i \in \mathcal{M}$.

Then expected costs are the followings:

Expected decision delay cost for $\delta = (\tau, d)$ is $\mathbb{E}[c(\tau - \theta)^+]$

Expected false alarm cost for $\delta = (\tau, d)$ is $\mathbb{E}[a_{0d}\mathbf{1}_{\{\theta \leq \tau < \infty\}}]$

Expected false isolation cost for $\delta = (\tau, d)$ is $\mathbb{E}[a_{\mu d}\mathbf{1}_{\{\theta \leq \tau < \infty\}}]$

For every decision $\delta = (\tau, d)$, we can define a Bayes risk function:

$$R(\delta) = c\mathbb{E}[(\tau - \theta)^+] + \mathbb{E}[a_{0d}\mathbf{1}_{\{\tau < \theta\}} + a_{\mu d}\mathbf{1}_{\{\theta \leq \tau < \infty\}}], \tag{3.3}$$

where $a_{ii} = 0$ for every $i \in \mathcal{M}$. The problem is to obtain $\delta = (\tau, d) \in \Delta$ (if it exists) with the *minimum Bayes risk*

$$R^* \triangleq \underset{\delta \in \Delta}{inf} R(\delta). \tag{3.4}$$

3.1 Optimal Stopping Problem

In this section, it is discussed that the Bayesian risk formulation (3.3) can be turned into a Markov optimal stopping problem [2].

Let $\Pi \triangleq \{\Pi_n = (\Pi_n^{(0)}, ..., \Pi_n^{(M)})\}_{n \geq 0}$ be a random process, Π_n^0 is the posterior probability that $\theta > n$ and Π_n^i is the posterior probability that $\theta \leq n$, $\mu = i$ with respect to the history of the first n observations of the process $X = (X_n)_{n \geq 1}$.

$$\Pi_n^{(0)} \triangleq \mathbb{P}\{\theta > n | \mathcal{F}_n\} \text{ and } \Pi_n^{(i)} \triangleq \mathbb{P}\{\theta \leq n, \mu = i | \mathcal{F}_n\},\ i \in \mathcal{M},\ n \geq 0. \tag{3.5}$$

Then the Bayes risk function can be rewritten as follows.

$$R(\delta) = \mathbb{E}\left[\sum_{n=0}^{\tau-1} c(1 - \Pi_n^{(0)}) + \mathbf{1}_{\{\tau < \infty\}} \sum_{j=1}^{M} \mathbf{1}_{\{d=j\}} \sum_{i=0}^{M} a_{ij}\Pi_\tau^{(i)}\right] \tag{3.6}$$

for a decision strategy $\delta = (\tau, d)$.

In [2], it is showed that the process $\Pi \triangleq \{\Pi_n = (\Pi_n^{(0)}, ..., \Pi_n^{(M)})\}_{n\geq 0}$ is a Markov process. Posterior probabilities are calculated as follows:

$$\Pi_{n+1}^{(i)} = \frac{D_i(\Pi_n, X_{n+1})}{D(\Pi_n, X_{n+1})},\ i \in \{0\} \cup \mathcal{M},\ n \geq 0, \tag{3.7}$$

$$\text{where} \quad D_i(\pi, x) \triangleq \begin{cases} (1-p)\pi_0 f_0(x) & \text{if } i = 0 \\ (\pi_i + \pi_0 p\nu) f_i(x) & \text{if } i \in \mathcal{M} \end{cases} \tag{3.8}$$

$$D(\pi, x) \triangleq \sum_{i=0}^{M} D_i(\pi, x).$$

Let us define functions $h, h_1, ..., h_M$: $\Pi \mapsto \mathbb{R}_+$

$$h(\pi) \triangleq \min_{j\in\mathcal{M}} h_j(\pi) \ \text{ and } \ h_j(\pi) \triangleq \pi_i a_{ij},\ j \in \mathcal{M}.$$

Then the minimum Bayes risk function R^* is turned into the below optimal stopping of Markov process Π:

$$R^* = \inf_{(\tau,d)\in\Delta} R(\tau, d) = \inf_{\tau\in\mathbb{F}} \mathbb{E}\left[\sum_{n=0}^{\tau-1} c\,(1 - \Pi_n^{(0)}) + \mathbf{1}_{\{\tau<\infty\}} h(\Pi_\tau)\right]. \tag{3.9}$$

The function $h_j(\Pi_n)$ calculates the expected cost of a decision $\delta = (\tau = n, d = j)$. It is always better to take the decision which gives minimum expected cost. Hence, if $h_i(\pi_n) = \min_{k\in\mathcal{M}} h_k(\pi_n)$, then the observer decides to raise an alarm at any stage n and claims that the change type is i. Since a_{ij} is known for all $(i, j) \in \mathcal{M}$, the observer gives the decision of isolation according to Π process at the time when an alarm is raised.

3.2 Optimality Equation

By using optimal stopping theory, Dayanık et al. [2] derive an optimal solution to the problem. The optimality equation is

$$V(\pi_n) = min\left\{h(\pi_n), g(\pi_n) + V(\pi_{n+1})\right\} \text{ and}$$

$$C(\pi_n) = \mathbb{E}[min\left\{h(\pi_{n+1}), g(\pi_{n+1}) + C(\pi_{n+1})\right\}],$$

where $g(\pi) = c(1 - \pi^{(0)})$. The optimal value for the problem is

$$R^* = V_0(1 - p_0, p_0\nu_1, \ldots, p_0\nu_M). \tag{3.10}$$

$g(\pi_n)$ is the expected cost of an observation when the system is at stage n. $C(\pi_n)$ represents the expected continuation cost at period n. Continuation cost is the cost incurred after the observer decides to continue and take one more observation.

Chapter 4

Background

4.1 An Overview of Dynamic Programming

The problems of decision making over time under uncertainty can be solved in principle with dynamic programming techniques. Sequential change diagnosis belongs to this class of problems.

Let us consider a system, whose state at time t is S_t. At each time t we make a decision x_t and then observe new exogenous information W_{t+1}, which together with x_t takes us from the state S_t to a new state S_{t+1}.

Making a decision x_t is rewarded with $C(S_t, x_t)$. The problem is to come up with a policy $\pi \in \Pi$ that solves

$$\sup_{\pi \in \Pi} \mathbb{E} \sum_{t=0}^{T} \gamma C(S_t, x_t),$$

where γ is a discount factor.

Suppose that at time t the system is in state S_t, an action x_t is taken, and $p(s|S_t, x_t)$ represents the probability that the system will be at state s at $t + 1$.

Then optimal decision is calculated as follows.

$$x_t = \arg\max_{x \in X} \left(C(S_t, x) + \gamma \sum_{s' \in S} p(s'|S_t, x) V_{t+1}(s') \right)$$

and

$$V_t(S_t) = \max_{x_t \in X} \left(C(S_t, x_t) + \gamma \sum_{s' \in S} p(s'|S_t, x_t) V_{t+1}(s') \right) \tag{4.1}$$

Equation 4.1 is called Bellman equation, which is also known as optimality equation. At each period of time, one can find the best action by solving the Bellman equation. However, to solve Bellman equation, one needs to know the value $V_{t+1}(s)$ of being in state s at the next period. One solution approach is the backward induction, which requires looping over all possible states. Suppose that the time horizon is finite, and the length of the horizon is T. Let S denote the set of all possible states. $V_T(s)$ for every state s is known. Assume that for some $1 \leq t \leq T$, $V_t(s)$ was calculated for all possible states $s \in S$. Given $V_t(s), V_{t-1}(s)$ can be calculated by Bellman equation for every $s \in S$. By applying this method backward in time, the value of being in any state $s \in S$ in any time t can be found. Hence, one can find the best action x by solving Bellman equation. However, the method is inapplicable to large problems due to curse of dimensionality. There are many challenges for the problems with multidimensional state variables, multidimensional random variables, and multidimensional decision variables. We can categorize computational difficulty of Bellman equation into three groups [3]:

1) Finding the value of being in a state

2) Computing expectation

3) Making decision

This three challenges are called three curses of dimensionality. First curse of dimensionality is due to state variable. If value of being in a state is a continuous variable, finding the value of being for all states is computationally infeasible. Even for the cases in which the state variable is discrete, if the state variable is a vector, then the number of possible states that one needs to calculate grows

exponentially. The second challenge is to compute the expectation in Bellman equation. It is difficult to compute the expectation if we do not know the probability distribution. The random variables can also be correlated. Finally, if the action variable is a vector, to solve the optimization problem in (4.1), one needs to enumerate the action space. However, like the first curse of dimensionality the number of possible actions grows exponentially and makes it impossible to enumerate. One way to overcome difficulties of dynamic programming is to discretize all the states and actions under the assumption that we can compute expectation. If the state and action spaces are not large, discretization can solve the problem. However, if the state variable or the action variable is vector-valued, then the problem still cannot be solved by Bellman equation.

4.2 An Overview of Approximate Dynamic Programming

In this section, a general discussion about approximate dynamic programming is given following [3]. Approximate dynamic programming is used to overcome the computational challenges of the classical dynamic programming [3]. As mentioned before, in dynamic programming, Bellman equation is generally solved by stepping backward in time. Suppose that time horizon is finite, and there are T periods to go. One needs to calculate value function for all possible values of the state variable S_t, starting with the time T. However, in approximate dynamic programming the basic idea is to step forward in time by simulating a sample path and to use an approximation for the value of being in a state at a future time. The main theme of approximate dynamic programming is to find a good approximation for the value function. In approximate dynamic programming we approximate the value function iteratively. We start with an initial approximation, denoted by $\bar{V}^0_{t+1}(s)$, for all states $s \in S$. Usually we assume that $\bar{V}^0_{t+1}(s) = 0$ for all states $s \in S$. Suppose we are in state S_t in iteration n and follow a sample path. To solve Bellman equation we need the value of being in state S_{t+1}. Instead of using true value of being in state S_{t+1}, we use the value

function approximation after $n-1$ iterations. The value function approximation after $n-1$ iterations is denoted by $\bar{V}_{t+1}^{n-1}(S_{t+1})$. Namely, in any period t of iteration n, $\bar{V}_{t+1}^{n-1}(S_{t+1})$ is used to compute value of being in state S_{t+1}. Then we determine x_t by solving Bellman equation.

$$x_t = \arg\max_{x \in X} \left(C(S_t, x) + \gamma \sum_{s' \in S} p(s'|S_t, x) \bar{V}_{t+1}^{n-1}(s') \right)$$

After the decision x_t, exogenous information W_{t+1} arrives. W_{t+1} comes from the sample path we follow. Given S_t, x_t and W_{t+1}, the next state S_{t+1} is computed. We proceed until the end of the time horizon. In each iteration, we update the value function approximation by performing a smoothing operation. We can not guarantee a good approximation by stepping forward through time following only one sample path. The idea is to repeat this procedure iteratively by using a fresh set of sample path realizations and update value function approximation as we progress. Simply, approximate dynamic programming is learning how to make decisions by simulating real life. A basic approximate dynamic programming algorithm is in Figure 4.1. Fundamental to approximate dynamic programming is to use an approximation for the value function. There are many ways to approximate the value function. A discussion about the value function approximation is given below.

4.2.1 Function Approximation

In Figure 4.1, it is assumed that we use a look-up table representation for the value function. That is, the state space is discretized into, say S^d elements, and we have an estimate $V_t(s)$ for each state $s \in S^d$. To have a good approximation for the value function, we iteratively update current estimate for value of being in a state. However, we update the value function approximation for only states which we actually visit. The problem about using look-up table representation is that we need an estimate for the value of being in each state that we may visit. For a large state space, the value approximation for only visited states does not

solve the problem. However, it replaces the computational difficulty in solving Bellman equation with the need for the value approximation for the states that may visit. The difficulty in using look-up table representation is to estimate one parameter for each state. That is, value function approximation is represented by using a large number of parameters. Estimating a large number of parameters introduces large statistical errors. Fundamental to approximate dynamic programming is to find a value function approximation which requires estimating smallest possible number of parameters. The general way of doing this is through state aggregation. The basic idea is to simplify state space by aggregating states into a smaller state space. Then the problem is solved by using aggregated state space. However, there is not a definitive way of deciding the correct level of aggregation. Coarser aggregation reduces the number of iterations the algorithm runs, but introduces errors. In the origins of dynamic programming, Bellman recognized that regression models can be used for value function approximation [17]. Using the vocabulary of approximation theory, expansion into basis functions is one of the most widely adopted strategy, which is simply a way of describing statistical regression models. The strategy is simply to use regression models for approximating the value function. Suppose that we use linear regression model, to estimate value function. We fit a model

$$V(S_t) = \beta_0 + \sum_i \beta_i \phi_i(S_t) + error$$

that estimates β vector given S_t vector. In our regression model, instead of using S_t values as independent variables we consider some functions of S_t values which are called basis functions and denoted by $\phi_i(S_t)$. One can think of the basis functions as the explanatory variables and $V(S_t)$ as the response variable. We regress the realizations of $V(S_t)$ on $\phi(S_t)$. The approximation of value function can be expressed in terms of basis function as follows.

$$V(S_t) \approx \bar{V}(S_t) = \beta_0 + \sum_i \beta_i \phi_i(S_t),$$

where β is the parameter vector and has to be estimated.

Step 0 Initialization:
Initialize $\bar{V}_t^0(S_t) = 0$ for all states $s \in S$.
Choose an initial state S_0^1.
Set $n = 1$.

Step 1 Choose a sample path ω^n.

Step 2 For $t = 0, 1, 2, ..., T$ do:
Solve:

$$\begin{aligned}\hat{v}_t^n &= \max_{x_t \in X} \left(C(S_t^n, x_t) + \gamma \mathbb{E}\{\bar{V}_{t+1}^{n-1}(S_{t+1})\}\right) \\ &= \max_{x_t \in X} \left(C(S_t^n, x_t) + \gamma \sum_{s' \in S} \mathbb{P}(s'|S_t^n, x_t) \bar{V}_{t+1}^{n-1}(s')\right),\end{aligned}$$

where $\mathbb{P}(S_{t+1}|S_t, x_t)$ is called transition matrix which is the probability that the next state will be S_{t+1} if we are in state S_t at time t and make a decision x_t.

Update $\bar{V}_t^{n-1}(S_t)$:

$$\bar{V}_t^n(S_t) = \begin{cases} (1 - \alpha_{n-1}) \bar{V}_t^{n-1}(S_t^n) + \alpha_{n-1} \hat{v}_t^n, & if \quad S_t = S_t^n, \\ \bar{V}_t^n(S_t), & otherwise. \end{cases}$$

Compute S_{t+1}^n by using S_t^n, x_t and W_{t+1}^n which is coming from ω^n.

Step 3 If $n = N$ stop.
Otherwise set $n \leftarrow n + 1$, go Step 1.

Figure 4.1: A basic approximate dynamic programming algorithm

Chapter 5

Methodology

5.1 Dynamic Programming Approach for Sequential Change Diagnosis Problem

Dayanık et al. [2] derive an optimality equation for the sequential diagnosis problem. We begin with the dynamic programming model for the sequential change diagnosis problem. For every stage s, the posterior probability vector $\Pi_s \triangleq (\Pi_s^0, \Pi_s^1, ..., \Pi_s^M)$ is the state variable. The state space which is the set of all possible state variables is the standard M-dimensional probability simplex S^M. The observer takes actions based on the state variable Π_s. At any period s, the observer decides between stopping and waiting for at least one more observation. The expected cost of stopping is $h(\Pi_s)$ and the expected cost of waiting for at least one more observation is $g(\Pi_s) + C(\Pi_s)$. The observer determines to stop and raise an alarm if $h(\Pi_s) < g(\Pi_s) + C(\Pi_s)$ and waits for at least one more observation if $h(\Pi_s) > g(\Pi_s) + C(\Pi_s)$. The continuation cost, $C(\Pi_s)$, is the cost which the observer will face at the next period, $s + 1$, if he decides to continue:

$$C(\pi_s) = \min\{h(\pi_{s+1}), g(\pi_{s+1}) + C(\pi_{s+1})\}. \tag{5.1}$$

$C(\pi_s)$ depends on the expected continuation cost of the next period. Hence, at any period s, before making a decision between stopping and waiting, the observer needs to know expected continuation cost of the next period. It can be easily seen that the state space, S^M increases exponentially in the number of alternative change distributions M. Hence, solving sequential change diagnosis problem via classical dynamic programming is computationally intractable due to curse of dimensionality.

5.1.1 Value Iteration

For finite horizon problems, value iteration algorithm is similar to backward dynamic programming. The difference is that we first initialize the value of being in any state and then update the value of being in that state iteratively until the convergence is satisfied [3]. The value iteration algorithm for sequential change diagnosis problem is in Figure 5.1. However, for large values of M, solving the problem via value iteration becomes intractable. Here, approximate dynamic programming steps in.

Step 0 Initialization:
Initialize $C^0(\pi) = 0$ for all states π in the discretized state space S_D^M.
Fix a tolerance parameter $\epsilon > 0$.
Set $n = 1$.

Step 2 For all $\pi \in S_D^M$ do:
Simulate next state π'
Compute:
$$C^n(\pi) = \min\{h(\pi''), g(\pi'') + C^{n-1}(\pi'')\}$$
where π'' is the nearest state to π' in S_D^M.

Step 3 If $\max_{\pi \in S_D^M} \left\| C^n(\pi) - C^{n-1}(\pi) \right\| < \epsilon$, stop.
Otherwise, set $n \leftarrow n + 1$, go Step 1.

Figure 5.1: Value Iteration

5.2 Approximate Dynamic Programming Approach for Sequential Change Diagnosis Problem

We begin by discussing our approximate dynamic programming algorithm at a basic level. We mention the details later. An overview of our approximate dynamic programming algorithm is as follows. We simulate $\Pi_0, \Pi_1, \ldots$ until we stop and raise an alarm. Details of simulation is discussed later. As we decide between stopping and waiting along the entire sample path, suppose that we use the same continuation function $C(\pi)$. If we have not stopped before time s yet, then at time s, we stop if $h(\pi_s) \leq C(\pi_s) + g(\pi_s)$ and wait if $h(\pi_s) > C(\pi_s) + g(\pi_s)$. When we eventually stop and raise an alarm, say after S periods of time, we learn a new continuation function. We have:

Time	State	Continuation Function Estimate
0	Π_0	$\min\{h(\Pi_1), g(\Pi_1) + C(\Pi_1)\} = \hat{C}_0$,
1	Π_1	$\min\{h(\Pi_2), g(\Pi_2) + C(\Pi_2)\} = \hat{C}_1$,
$\vdots$	$\vdots$	$\vdots$
$S-1$	Π_{S-1}	$\min\{h(\Pi_S), g(\Pi_S) + C(\Pi_S)\} = \hat{C}_{S-1}$.

Then we can apply a function approximation technique to the data $(\Pi_i, \hat{C}_i), i = 0 \ldots, S-1$. However, when we stop and raise an alarm, we may not have sufficient observations to learn a new continuation function. A remedy to this problem is to repeat above process with a fresh set of sample realizations until at least a specified number of observations, say $\tilde{S}$, are accumulated. After we eventually collect sufficient number of observations, we use these data to fit a regression model. We decided to approximate the continuation function with an affine combination of radial basis penalized splines. Suppose that $\hat{\beta}$ is the vector of parameters of the regression. Due to random sampling of the sample paths, there is some noise in $\hat{\beta}$. Hence, we perform a smoothing operation to find the current estimate of the parameter vector, denoted by $\bar{\beta}$ by the formula

$$\bar{\beta}^n = (1-\alpha)\hat{\beta}^n + \alpha\bar{\beta}^{(n-1)}, \tag{5.2}$$

where $\alpha \in (0,1)$ is a fixed smoothing parameter, and n is the iteration number. Figure 5.2 gives the details of the algorithm.

Step 0 Initialization:
Initialize $\bar{\beta}^0 = 0$.
Fix a tolerance parameter $\epsilon > 0$.
Fix the number of required observations for function approximation $\tilde{S}$.
Set $n = 1$.
Set $\tilde{s} = 0$.

Step 1 Choose a sample path.
Set $s = 1$.
Set initial state $\pi_1^n = (\pi_1^{n,0}, \ldots, \pi_1^{n,M})$ where $\pi_1^{n,0} = 1 - p_0$ and $\pi_1^{n,i} = p_0 \nu_i$.

Step 2 Compute:

$$\hat{C}_s^n(\pi_s^n) = \min\left\{h(\pi_{s+1}^n), g(\pi_{s+1}^n) + \bar{C}^{n-1}(\pi_{s+1}^n)\right\}$$

where π_{s+1}^n is the next state on the sample path.

Step 3 If $\hat{C}_s^n(\pi_s^n) + g(\pi_s^n) < h(\pi_s^n)$, set $s \leftarrow s + 1$ and go Step2.
Otherwise, set $\tilde{s} \leftarrow \tilde{s} + s$,
if $\tilde{s} < \tilde{S}$, go step1,
otherwise, go step 4.

Step 4 Update $\bar{\beta}^{n-1}$:
Apply a function approximation technique to the data and find $\hat{\beta}^n$.
$\bar{\beta}^n = (1 - \frac{1}{n})\beta^{n-1} + \frac{1}{n}\hat{\beta}^n$.
Set $\tilde{s} = 0$.

Step 5 If $\left\|\bar{\beta}^{n-1} - \bar{\beta}^n\right\| < \epsilon$ stop.
Otherwise set $n \leftarrow n + 1$, go Step 1.

Figure 5.2: Approximate Dynamic Programming Algorithm

5.2.1 Simulating Posterior Probability Process Π

In Figure 5.2, at any period s of iteration n, a decision between stopping and waiting for at least one more observation is taken. Suppose that we are in period $s + 1$ and we decide to take one more observation. $\Pi_0, \Pi_1, ..., \Pi_s$ are already calculated based on the past and present observations. We can, therefore simulate

the next observation, X_{s+1}, according to

$$\mathbb{P}\{X_{s+1} \in dx|\pi_0, ..., \pi_s\} = \mathbb{P}\{X_{s+1} \in dx|\pi_s\}\pi_s^{(0)}(1-p)f_0(x)dx + \sum_{j=1}^{M} \pi_s^{(0)} p\nu_j f_j(x)dx + \sum_{j-1}^{M} \pi_s^{(j)} f_j(x)dx.$$

Then we calculate Π_{s+1} by (3.7). Figure 5.3 describes the simulation algorithm.

Suppose that we have not raised an alarm at or before time s, and $\pi_0, \ldots \pi_s$ are already calculated.

Simulate X_{s+1}:
There are three cases:
Case 1. Change occurs before time s.
Case 2. No change occurs before time $s+1$.
Case 3. Change occurs at time s.

Simulate a uniformly distributed random number u_1 in the interval $[0,1]$

If $\sum_{i=1}^{j-1} \pi_s^i \leq u_1 < \sum_{i=1}^{j} \pi_s^i$ for some $j \in \mathcal{M}$, then simulate X_{n+1} from f_j. (Case 1)

Else, simulate a uniformly distributed random number u_2 in the interval $[0,1]$.

If $p < u_2 \leq 1$ then simulate X_{n+1} from f_0. (Case 2)

If $0 < u_2 \leq p$ then simulate a uniformly distributed random variable u_3 in the interval $[0,1]$.

Simulate X_{n+1} from f_l where $\sum_{k=0}^{l-1} \nu_k \leq u_3 \leq \sum_{k=0}^{l} \nu_k$, $l \in \mathcal{M}$. (Case 3)

Figure 5.3: Simulation of the next observation

5.2.2 Value Function Approximation

To approximate the expected continuation cost function, we use regression methods which we overviewed in Chapter 4. In each iteration, we collected all observed state variables and the corresponding continuation cost estimates until we finally raised an alarm. We approximate the continuation cost function by using this collection of data. If we expected that the continuation cost $C(\pi)$ were affine function of the state π, then we would have fitted a multiple linear regression

model

$$C(\pi) = \beta_0 + \sum_i \beta_i \pi_i + error,$$

where β would be estimated by the least squares estimate $\hat{\beta}$, and the continuation cost would be estimated by $\hat{C}(\pi) = \hat{\beta}_0 + \sum_i \hat{\beta}_i \pi_i$. However, $C(\pi)$ is a nonlinear concave function of π, and we model $C(\pi)$ as an affine combination of some suitable basis functions, denoted by $\phi(\pi)$. One can think of the basis functions as the feature/explanatory variables. Let I denote the number of basis functions in the regression model. We regress the realizations of the continuation cost, $\hat{C}(\pi_s)$ on basis functions, $\phi_i(\pi_s)$ where $i = 1, ..., I$. The approximation of continuation cost function can be expressed in terms of basis function as

$$\begin{aligned} C(\pi) &= \beta_0 + \sum_i \beta_i \phi_i(\pi) + error, \\ \mathbb{E}\left[C(\pi)\right] = \bar{C}(\pi) &= \beta_0 + \sum_i \beta_i \phi_i(\pi), \\ \bar{C}(\pi) \approx \hat{C}(\pi) &= \hat{\beta}_0 + \sum_i \hat{\beta}_i \phi_i(\pi), \end{aligned} \tag{5.3}$$

where $\hat{\beta}$ is the least squares estimate of the parameter vector β. We decided to use the very flexible radial basis functions in the regression models. A general discussion about these regression models is given below following [18]:

5.2.2.1 Radial Basis Function Expansions

Consider the simple linear regression model,

$$\mathbf{y} = X\boldsymbol{\beta} + \epsilon$$

where $\mathbf{y}=\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$, $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$. The columns of $\mathbf{X}$ matrix is called basis functions for the regression model. Note that $\mathbf{y}$ is the linear combination of columns of $\mathbf{X}$ matrix. A linear regression model can easily be fitted to the data in Figure 5.4. However, it is not possible to model the data in 5.5 by using 1

and x as basis functions since the data contain nonlinear effects. On the other hand, the basis functions $1, x$, and $(x - 0.6)_+$ can handle the nonlinearity in the data. Those two examples show that the selection of the basis functions affects the quality of regression model to estimate the response variable.

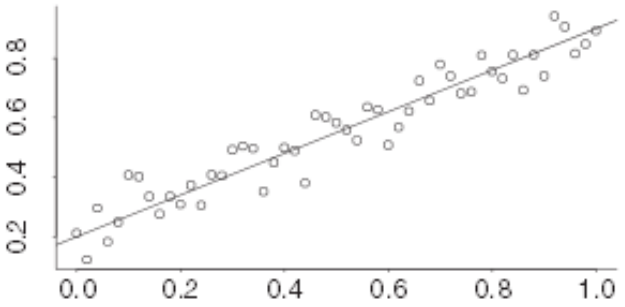


Figure 5.4: Linear Regression Model

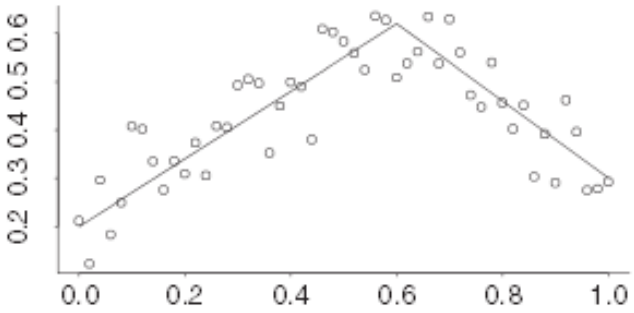


Figure 5.5: Broken Stick Model

Radial basis functions are in the form of $\|x - \kappa_k\|^p$, where κ_k is a knot point and p is an odd number. The model consists of K knot points $\kappa_1, \ldots, \kappa_K$. Consider the model,

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_{m-1} x_i^{m-1} + \sum_{k=1}^{K} \beta_{mk} \|x_i - \kappa_k\|^{2m-1}, \qquad (5.4)$$

where $m = \frac{p+1}{2}$.

Figure 5.6 shows the basis functions corresponding to $p = 1$.

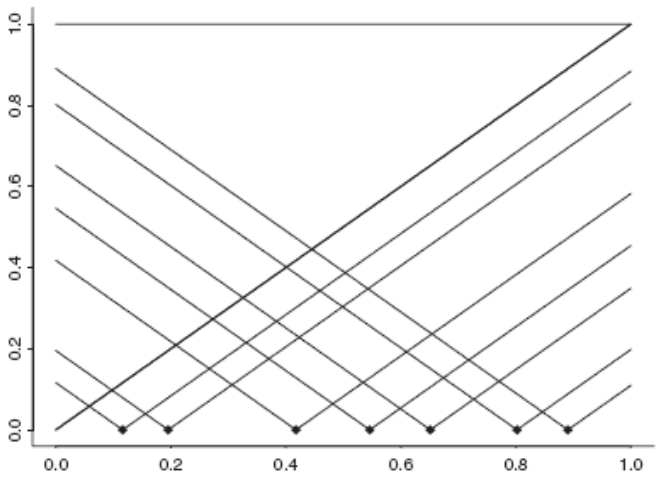


Figure 5.6: Basis Functions Corresponding to $p = 1$

The advantage of using radial basis functions is that the extension of the model to higher-dimensional predictor variables is simple. Suppose that $\mathbf{x} \in \mathbb{R}^d$ and $\kappa_1, \ldots, \kappa_K$ are knots in $\mathbb{R}^d$, then basis functions are $\|\mathbf{x} - \kappa_\mathbf{k}\|^p$, where $\|.\|$ is the Eucledian distance in $\mathbb{R}^d$. This is useful since the $M + 1$ dimensional state variable is the predictor variable in our model. Any complex data can be easily handled by using radial-basis functions. The number of knots in the model is crucial since too many knots result in overfitting and too few knots introduce bias. A penalized spline regression method can help continuously control the smoothness of the fitted model. In penalized spline regression, large number of knots is preferred, but the model complexity is penalized. The logic is to limit the effects of some knots in the model. Consider the radial basis penalized spline model with K knots. Assume that K is a large number. The minimization problem is

$$\min \|\text{y-X}\beta\|^2, \tag{5.5}$$

where $\beta = [\beta_0, \ldots, \beta_{m-1}, \beta_{m1}, \ldots, \beta_{mK}]^T$ and β_{mk} is the coefficient of the k^{th} knot. Unconstrained estimation of the β_{mk} results in overfitting. An easy to implement constraint is that $\sum \beta_{mk}^2 < C$ for some number C. For a penalty matrix $\mathbf{K} = \left[\|\kappa_k - \kappa_k\|^{2m-1}_{1 \le k,k \le K} \right]$, the minimization problem is

$$\min \|\text{y-X}\beta\|^2 \text{ subject to } \beta^\mathbf{T}\text{K}\beta \le \mathbf{C}.$$

Via Lagrange multiplier rule, the above minimization problem becomes

$$\min \|\text{y-X}\beta\|^2 + \lambda^{2m-1}\beta^{\mathrm{T}}\mathbf{K}\beta$$

where $\mathbf{X}=\begin{bmatrix} 1 & x_1 \dots & x_1^{m-1} & |x_1-\kappa_1|^{2m-1} & \dots & |x_1-\kappa_K|^{2m-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n \dots & x_n^{m-1} & |x_n-\kappa_1|^{2m-1} & \dots & |x_n-\kappa_K|^{2m-1} \end{bmatrix}$

The solution of the minimization problem is

$$\hat{\beta}_\lambda = \left(\mathbf{X^T X} + \lambda^{2m-1}\mathbf{K}\right)^{-1}\mathbf{X^T y}.$$

Fitted values are

$$\hat{\mathbf{y}} = \mathbf{X}\left(\mathbf{X^T X} + \lambda^{2m-1}\mathrm{K}\right)^{-1}\mathbf{X^T y}.$$

One important issue is how to choose the knots. Selecting knots without considering the distribution of the predictor space results in waste of knots. One approach to overcome this problem gives the knot locations according to

$$\kappa_k = (k/K)^{th}$$

sample quantile of $x_1, \dots, x_n$. Sample quantiles correspond to maximal separation of K points among the unique x_i . However, this solution works for one-dimensional predictor variable. Recall that, for our problem we are interested in high-dimensional smoothing. For high-dimensional data, space filling designs [19] correspond to maximal separation principle. To choose the knots we apply the procedure described in [18] with minor changes:

(1) The number of knots $K = \min\{\max\{20, \text{sample size}/4\}, 50\}$.

(2) If sample size < 1500 every explanatory variable, x_i, is a candidate to be a knot. Otherwise, take a random sample of the x_i of size 1500 as the candidate points to be a knot.

(3) Apply space filling algorithm to the candidate points. The space filling algorithm we use is the swapping algorithm [20], which is available in R software,

fields package. The swapping algorithm finds a set of points called design points among a given set of candidate set with respect to a space-filling criterion specified by the user. The criterion we used in our algorithm is the following. Our algorithm selects the design points D from the set C of the candidate points so as to minimize the criterion

$$f(C, D) = \left\| \sum_{c \in C} \left(\sum_{d \in D} \|d - c\|^p \right)^{q/p} \right\|^{1/q},$$

where p and q are some parameters. The number p affects the weight of distance from a point $c \in C$ to all points in D, and q affects the weight of the distance from all points in C to all points in D. Design points, assigned by the swapping algorithm among candidate points, are selected as the knots.

5.2.2.2 Linear Mixed Models

Linear mixed models are extensions of regression models which allow for both fixed and random effects. They are generally used to model grouped data. Fixed effects are population averaged parameters which influence the mean of the response. Random effects take into account the randomness due to sampling from different groups of data [18].

Linear mixed models can be illustrated at best with an example. Consider the scatterplot in Figure 5.7 consisting of weight measurements of 48 pigs for 9 successive weeks. Measurements belonging to the same pig are connected with a line. Let $weight_{ij}$ be the weight of pig i on week number j. The ordinary least squares model is as follows:

$$weight_{ij} = \beta_0 + \beta_1 week_j + \epsilon_{ij}, \;\; 1 \leq i \leq 48 \;\text{ and }\; 1 \leq j \leq 0, \tag{5.6}$$

with ϵ_{ij} i.i.d. $N(0, \sigma^2)$.
Figure 5.7 shows that weight measurements have less variability within each pig. To incorporate the variance among weights of different pigs, a random term U_i

to the least squares model is added. Then the model becomes

$$weight_{ij} = \beta_0 + U_i + \beta_1 week_j + \epsilon_{ij}. \tag{5.7}$$

The model allows different intercepts for different pigs. U_i represents intercept for the i^{th} pig, where $U_1, \ldots, U_{48}$ is a random sample from a $N(0, \sigma_U^2)$ distribution for some σ_U^2.

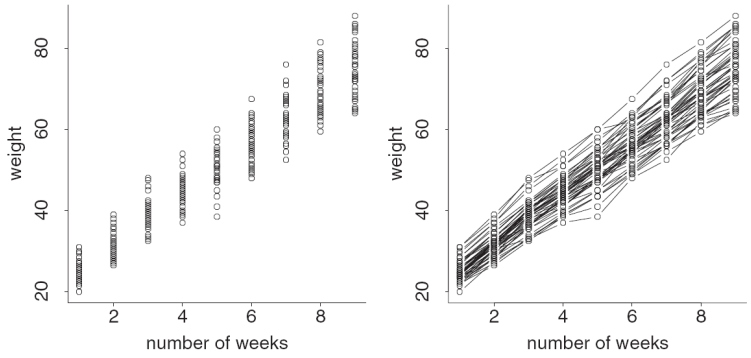


Figure 5.7: Pig Weight Measurements Data

For the pig example above, the fixed effects are β_0 and β_1, and the random effects are $U_1, \ldots, U_{48}$, and the covariance matrix parameters are σ_U^2 and σ_ϵ^2.

The standard linear model can be expanded to a linear mixed model as follows:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{u} + \epsilon, \tag{5.8}$$

$$\text{where } \mathbb{E}\begin{bmatrix}\mathbf{u} \\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix} \text{ and } \text{Cov}\begin{bmatrix}\mathbf{u} \\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}\end{bmatrix}.$$

$$\mathbf{V} \equiv \text{Cov}(\mathbf{y}) = \mathbf{Z}\mathbf{G}\mathbf{Z}^T + \mathbf{R}.$$

Here, $\mathbf{Z}$ is the design matrix for random effects, and $\mathbf{u} \sim N(0, \sigma_u^2), \boldsymbol{\epsilon} \sim N(0, \sigma_\epsilon^2)$. Pig example is a special case of linear mixed models, for which

$$\mathbf{y}=\begin{bmatrix} weight_{1,1} \\ \vdots \\ weight_{1,9} \\ \vdots \\ weight_{48,1} \\ \vdots \\ weight_{48,9} \end{bmatrix}, \quad \mathbf{X}=\begin{bmatrix} 1 & week_1 \\ \vdots & \\ 1 & week_9 \\ \vdots & \\ 1 & week_1 \\ \vdots & \\ 1 & week_9 \end{bmatrix}, \quad \boldsymbol{\beta}=\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix},$$

$$\mathbf{Z}=\begin{bmatrix} \mathbf{1}_{9\times 1} & \mathbf{0}_{9\times 1} & \dots & \mathbf{0}_{9\times 1} \\ \mathbf{0}_{9\times 1} & \mathbf{1}_{9\times 1} & \dots & \mathbf{0}_{9\times 1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{9\times 1} & \mathbf{0}_{9\times 1} & \dots & \mathbf{1}_{9\times 1} \end{bmatrix}, \quad \mathbf{u}=\begin{bmatrix} U_1 \\ \vdots \\ U_{48} \end{bmatrix},$$

$$\mathbf{G}=\sigma_U^2\mathbf{I} \quad \text{and} \quad \mathbf{R}=\sigma_\epsilon^2\ \mathbf{I}.$$

Consider the model in (5.4). Let

$$\beta=\begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{m-1} \end{bmatrix} \text{ and } u=\begin{bmatrix} \beta_{m,0} \\ \vdots \\ \beta_{m,K-1} \end{bmatrix},$$

$$X=\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{m-1} \end{bmatrix},$$

$$Z=\begin{bmatrix} |x_1-\kappa_0|^{2m-1} & \dots & |x_1-\kappa_{K-1}|^{2m-1} \\ \vdots & \ddots & \vdots \\ |x_n-\kappa_0|^{2m-1} & \dots & |x_n-\kappa_{K-1}|^{2m-1} \end{bmatrix}.$$

Then the radial basis penalized splines model can be viewed as a linear mixed model

$$y = X\beta + Zu + \epsilon. \tag{5.9}$$

We need to estimate $\boldsymbol{\beta}$, predict $\mathbf{u}$, and estimate the parameters in $\mathbf{G}$ and $\mathbf{R}$. One of the ways to estimate the parameters in the model is to use best linear unbiased prediction method. Consider the linear model

$$\text{y=X}\beta + \epsilon^* \text{ with correlated errors } \boldsymbol{\epsilon}^* = Zu + \epsilon \text{ where } \text{Cov}(\boldsymbol{\epsilon}^*) = \mathbf{V}.$$

For a given $\mathbf{V}$,

$$\hat{\boldsymbol{\beta}} = \mathbf{X^T}(\mathbf{V^{-1}X})^{-1}\mathbf{X^T}\mathbf{V^{-1}}\mathbf{V^{-1}y}$$

and for a given $\hat{\boldsymbol{\beta}}$,

$$\hat{\mathbf{u}} = \mathbf{GZ^T}\mathbf{V^{-1}}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})$$

are best linear unbiased predictors (BLUP) of $\boldsymbol{\beta}$ and $\mathbf{u}$. As mentioned before, mixed models and penalized splines have close ties. Indeed, penalized spline estimation can be written as BLUP of a mixed model. BLUP representation of the radial basis penalized spline model is as follows:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{m-1} \end{bmatrix},$$

$$\mathbf{Z} = \begin{bmatrix} |x_1 - \kappa_0|^{2m-1} & \dots & |x_1 - \kappa_{K-1}|^{2m-1} \\ \vdots & \ddots & \vdots \\ |x_n - \kappa_0|^{2m-1} & \dots & |x_n - \kappa_{K-1}|^{2m-1} \end{bmatrix},$$

$$\mathbf{G} = \sigma_u^2\mathbf{I}, \quad \mathbf{R} = \sigma_\epsilon^2\mathbf{I}, \quad \lambda_{2m-1} = \sigma_\epsilon^2/\sigma_u^2.$$

5.2.3 Updating Value Function Approximation

Recall that, in each iteration we fit a regression model to learn the continuation function. $\hat{\beta}^n$ is the parameter vector of the regression model at iteration n. Then we perform the smoothing operation:

$$\bar{\beta}^n = (1-\alpha)\hat{\beta}^n + \alpha\bar{\beta}^{n-1}.$$

Since we select knots among predictor variables by using a space filling algorithm, the regression model changes in each iteration with respect to the selected knots. Consequently, $\hat{\beta}^n$ and $\bar{\beta}^{n-1}$ are the parameters of different regression models, and we can not perform the smoothing operation above. Suppose that we are at iteration n. We perform smoothing as follows:

Step 1. Generate predictor variables, $\tilde{x_i}$'s.

Step 2. Find corresponding response variables, $\tilde{y_i}$'s by using the regression model of previous iteration $n-1$.

Step 3. Find parameters of the regression model:

$$\tilde{y_i} = \tilde{\beta}_0 + \tilde{\beta}_1\tilde{x}_i + \tilde{\beta}_2\tilde{x}_i^2 + \ldots + \tilde{\beta}_{m-1}\tilde{x}_i^{m-1} + \sum_{k=1}^{K}\tilde{\beta}_{mk}\left\|\tilde{x}-\kappa_k\right\|^{2m-1} + error_i,$$

where κ_k's are knots of the regression model of current iteration n.

Step 4. Perform smoothing:

$$\bar{\beta} = (1-\alpha)\tilde{\beta} + \alpha\hat{\beta}.$$

5.2.4 Exploration vs. Exploitation

Making the best decision based on current estimate of the value function is called a greedy strategy [3]. In a greedy strategy, some states might look worse since we

have not visited them and we never visit and discover these states. We may loop over the visited states and stuck in a local optimum. This problem is referred as the exploration vs. exploitation problem. Exploration stands for visiting states randomly for just discovering new states. However, for problems with large state spaces, making decision randomly can be costly and does not ensure that we visit all the states. Hence, at some point we should exploit from our current estimate and make decision based on it. One of the challenges in approximate dynamic programming is to strike balance between exploration and exploitation. There are some mix strategies. For example, an exploration rate ρ can be specified. With probability ρ we decide randomly and with probability $1 - \rho$ we take the best action based on current estimate of value function. The results of mixed strategies we used are demonstrated in the computational study section.

Chapter 6

Computational Study

One of the challenges of approximate dynamic programming is to decide how to evaluate a new algorithm. Powell [21] states that it is essential to have some sort of benchmark. One of the strategies suggested is to compare the algorithm to an optimal Markov decision process. The strategy is to simplify the complex problem and solve it optimally via dynamic programming. Then we solve the simpler problem by using the ADP algorithm and compare the results. To evaluate our algorithm we solve the sequential change diagnosis problem optimally for the special cases $M = 2$ and $M = 3$. We apply our ADP algorithm to the same examples. The results of the approximate and optimal algorithm are then compared. We used the examples of [2] for comparison.

6.1 Two Alternatives After the Change

In this section we study the examples in which there are two change distributions, $f_1(.)$ and $f_2(.)$. We use the value iteration and approximate dynamic programming algorithms to find optimal and approximate continuation costs and optimal and approximate decision regions over a fine discretization of S^2. A state π is in the stopping region if $h(\pi) < C(\pi) + g(\pi)$ and in the continuation region otherwise.

Graphical representation of stopping and continuation regions is illustrated by using linear mapping $L : \mathbb{R}^3 \to \mathbb{R}^2$, defined by $L(\pi_0, \pi_1, \pi_2) \triangleq \left(\frac{2}{\sqrt{3}}\pi_1 + \frac{1}{\sqrt{3}}\pi_2 + \pi_2\right)$. Two dimensional probability simplex, $S^2 \in \mathbb{R}^3$ can be mapped into $\mathbb{R}^2$ via $L(\pi)$ (see Figure 6.1).

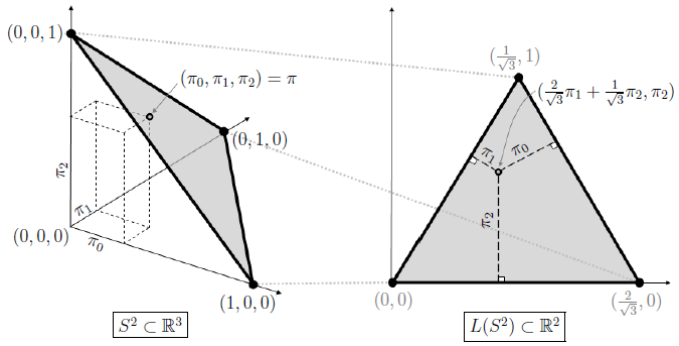


Figure 6.1: Linear Mapping from $S^2 \in \mathbb{R}^3$ to $L(S^2) \in \mathbb{R}^2$

We study six examples with different cost parameters as depicted by Table 6.1. For all six examples, the change distributions $f_0 = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), f_1 = \left(\frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}\right), f_2 = \left(\frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}\right)$ and the parameters, $p_0 = \frac{1}{50}$, $p = \frac{1}{20}$, $\nu_1 = \nu_2 = \frac{1}{2}$ are in common.

Example	a_{01}	a_{02}	a_{12}	a_{21}	c
1	10	10	3	3	1
2	50	50	3	3	1
3	10	10	10	10	1
4	10	10	16	4	1
5	14	20	8	1	1
6	14	20	8	1	2

Table 6.1: Model Parameters

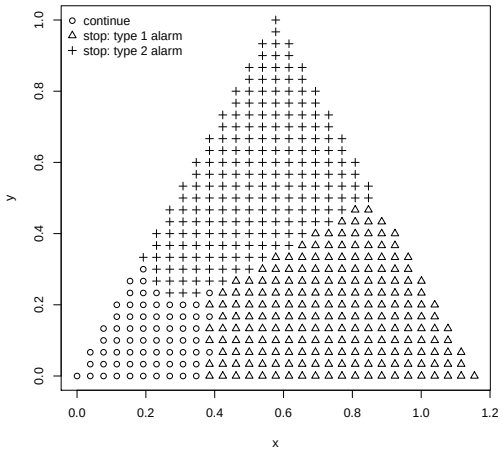
Figure 6.2(a) illustrates optimal decision regions obtained by the value iteration for Example 1. Figures 6.2(b) and 6.2(c) show the decision regions of ADP algorithms for pure exploitation strategy and mixed strategy with exploration rate $\rho = 0.5$, respectively. For a fine discretization of S^2, we compare optimal continuation continuation cost and the continuation cost of ADP algorithm. Let S_d^2 be the discretized state space. Let $C(\pi)$ be the continuation for state π computed by using the value iteration algorithm, and $\bar{C}(\pi)$ be the continuation cost approximated by ADP. Figures 6.3(a) and 6.3(b) are the histograms of percentage

relative error $\frac{C(\pi)-\bar{C}(\pi)}{C(\pi)} \times 100$ where $\pi \in S_d^2$ for pure exploitation strategy and for mixed strategy with exploration rate $\rho = 0.5$. The similarity of optimal and approximated decision regions shows the performance of ADP algorithm. Decision regions in Figures 6.2(b) and 6.2(c) are almost identical to optimal decision regions in 6.2(a). The distribution of the percentage relative error is also a performance criterion. We expect that the percentage relative error is distributed around zero. In 6.3(a) and 6.3(b), percentage relative error is dense around zero. Also, in Examples 2–6, approximated decision regions are similar to the optimal decision regions (see Figures 6.2, 6.4, 6.6, 6.8, 6.10 and 6.12) and percentage relative error is accumulated highly around zero (see Figures 6.3, 6.5, 6.7, 6.9, 6.11and 6.13).

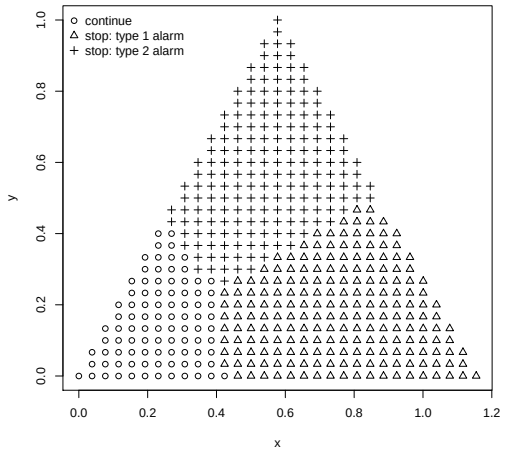
Both pure exploitation and mixed strategies provide reasonable approximation. Since the state space is not very large for $M = 2$, we can still explore most of the states with pure exploitation strategy. Therefore, the difference between decision regions in pure exploitation strategy and mixed strategy is not remarkable.

6.2 Three Alternatives After the Change

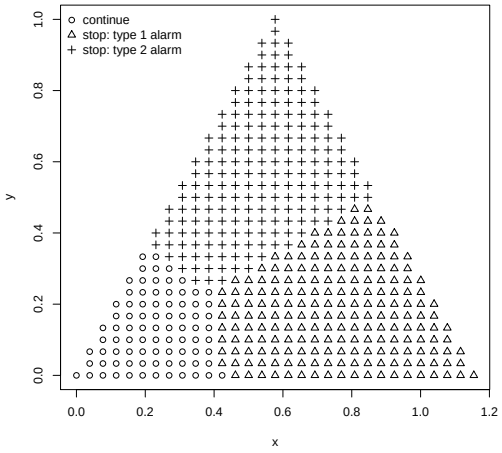
In this section we study the case $M = 3$ in which there are three change distributions, $f_1(.)$, $f_2(.)$ and $f_3(.)$. Similar to the two alternative change section, we generate the optimal and approximate decision regions mapped into S^3 by the linear mapping $L : \mathbb{R}^4 \to \mathbb{R}^3$ is defined by $L(\pi_0, \pi_1, \pi_2, \pi_3) \triangleq \left(\sqrt{\frac{3}{2}}\pi_1 + \frac{1}{2}\sqrt{\frac{3}{2}}\pi_2, \frac{1}{2}\sqrt{\frac{3}{2}}\pi_3, \frac{3}{2}\sqrt{\frac{1}{2}}\pi_2 + \frac{1}{2}\sqrt{\frac{1}{2}}, \pi_3\right)$. The change distributions are $f_0 = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), f_1 = \left(\frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}\right), f_2 = \left(\frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}\right), f_3 = \left(\frac{3}{10}, \frac{2}{10}, \frac{2}{10}, \frac{3}{10}\right)$. The parameters are $c = 1$, $a_{0j} = 40$, $a_{ij} = 20$ $i, j = 1, 2, 3$. Figure 6.14 shows the histogram of percentage difference for pure exploitation and exploration rate $\rho = 0.5$.



(a) Optimal decision regions

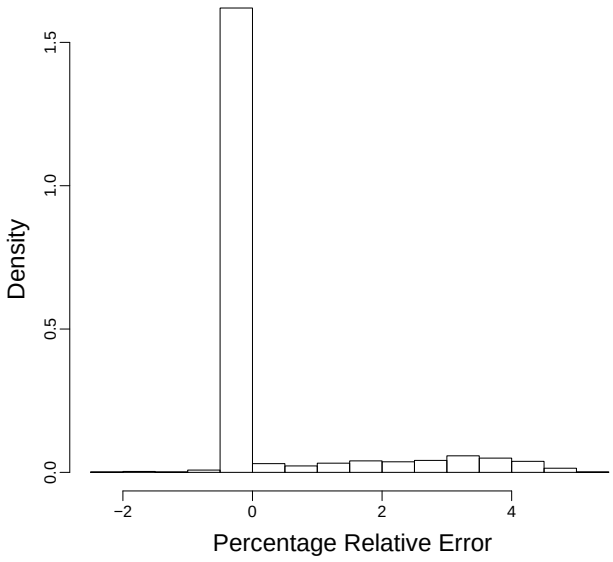


(b) Decision regions of the ADP algoritm with pure exploitation

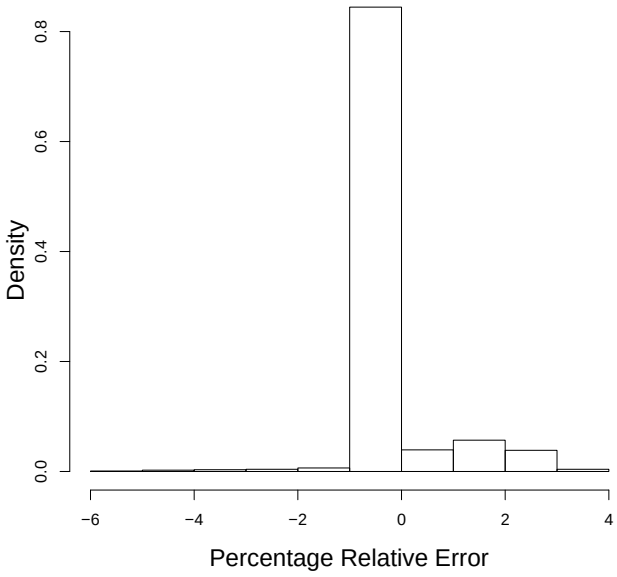


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.2: Decision Regions for Example 1: $a_{01} = a_{02} = 10, a_{12} = a_{21} = 3, c = 1$

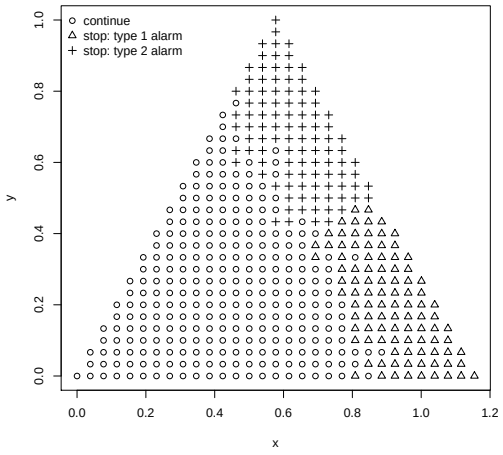


(a) Percentage relative error of the ADP algorithm with pure exploitation

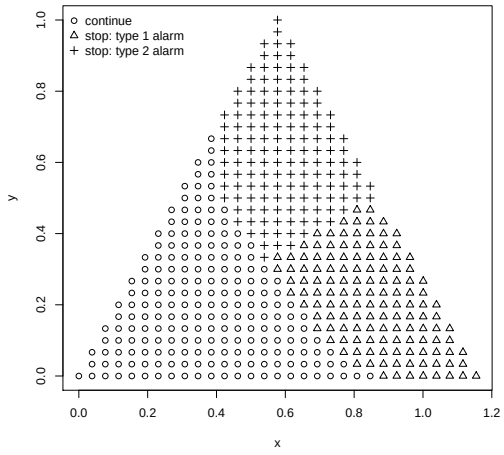


(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

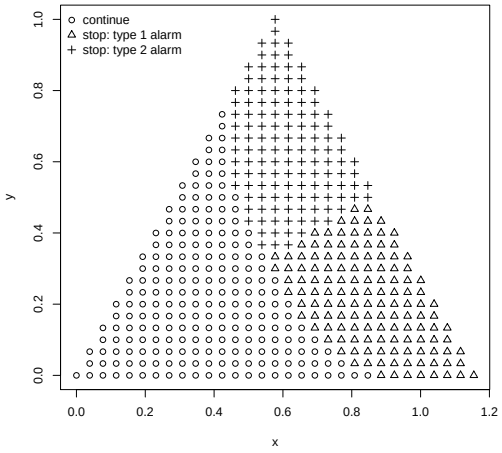
Figure 6.3: Histograms of Percentage Relative Error for Example 1: $a_{01} = a_{02} = 10, a_{12} = a_{21} = 3, c = 1$



(a) Optimal decision regions

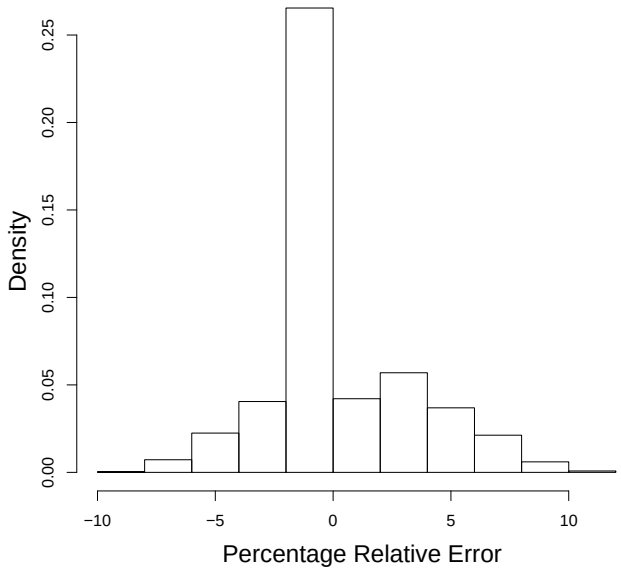


(b) Decision regions of the ADP algoritm with pure exploitation

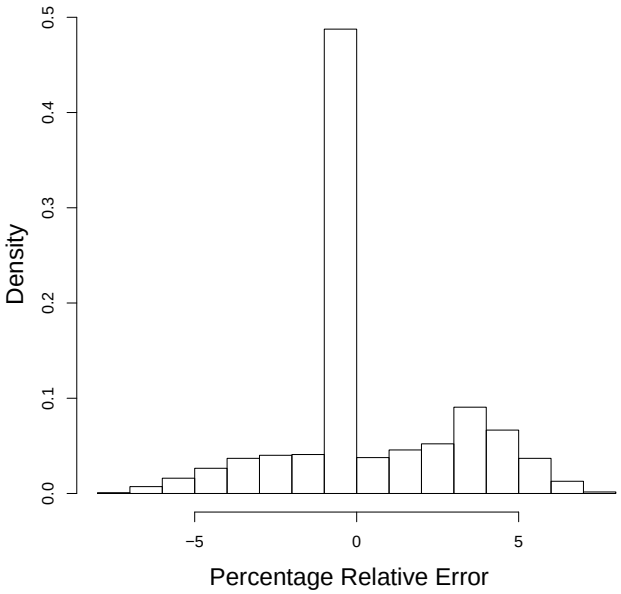


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.4: Decision Regions for Example 2: $a_{01} = a_{02} = 50, a_{12} = a_{21} = 3, c = 1$

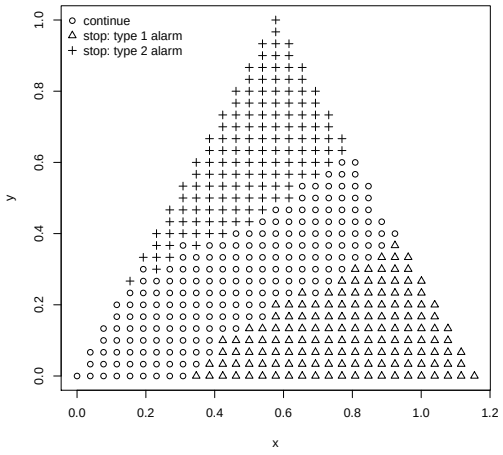


(a) Percentage relative error of the ADP algorithm with pure exploitation

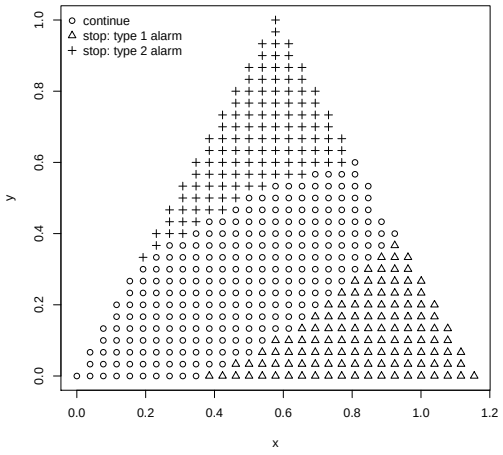


(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

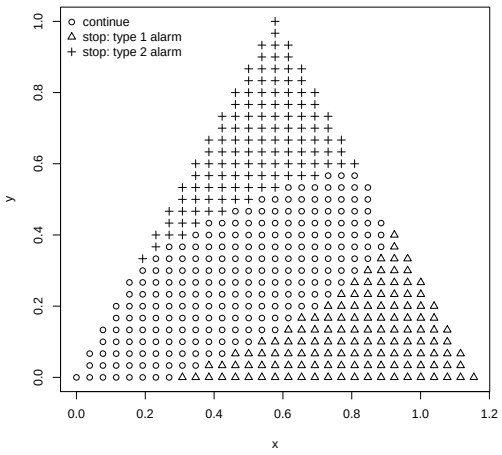
Figure 6.5: Histograms of Percentage Relative Error for Example 2: $a_{01} = a_{02} = 50, a_{12} = a_{21} = 3, c = 1$



(a) Optimal decision regions

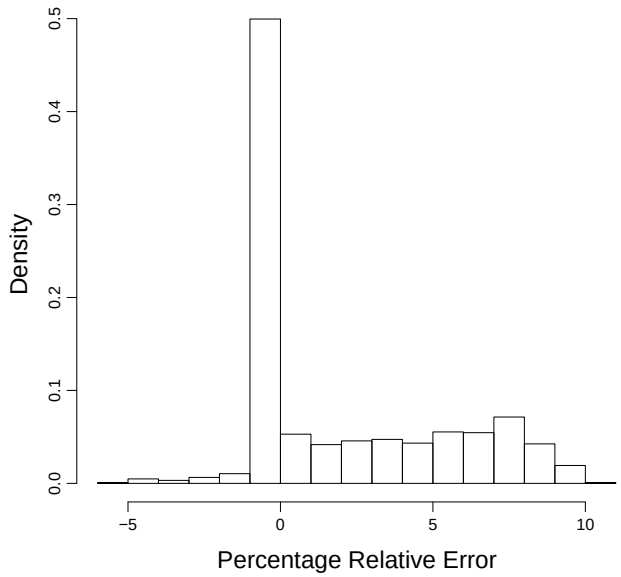


(b) Decision regions of the ADP algoritm with pure exploitation

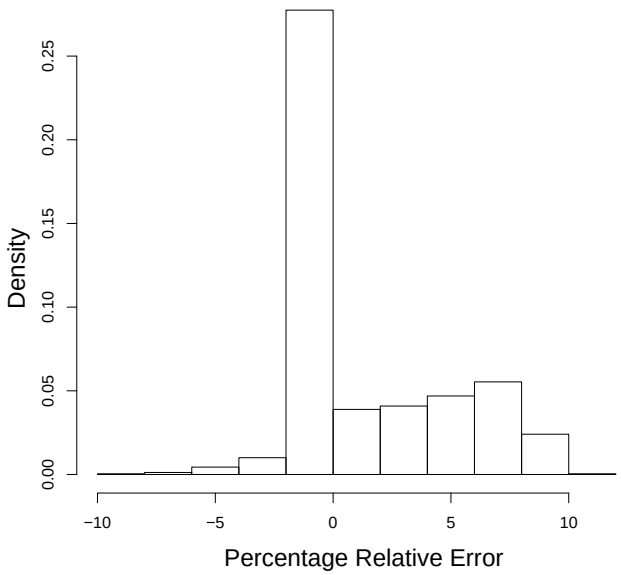


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.6: Decision Regions for Example 3: $a_{01} = a_{02} = 10, a_{12} = a_{21} = 10, c = 1$

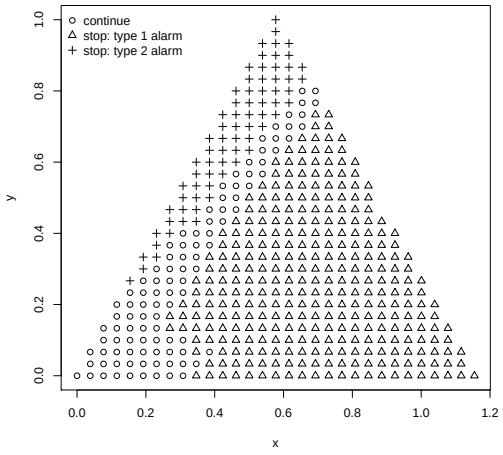


(a) Percentage relative error of the ADP algorithm with pure exploitation

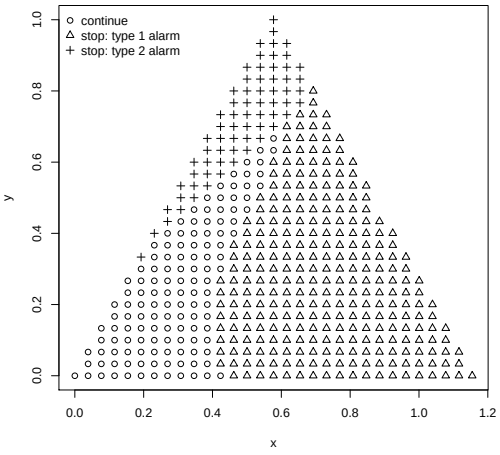


(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

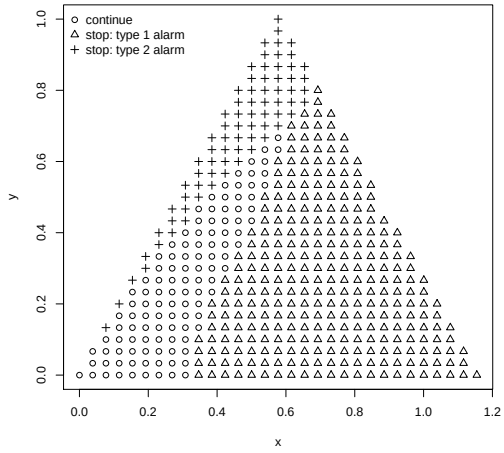
Figure 6.7: Histograms of Percentage Relative Error for Example 3: $a_{01} = a_{02} = 10, a_{12} = a_{21} = 10, c = 1$



(a) Optimal decision regions

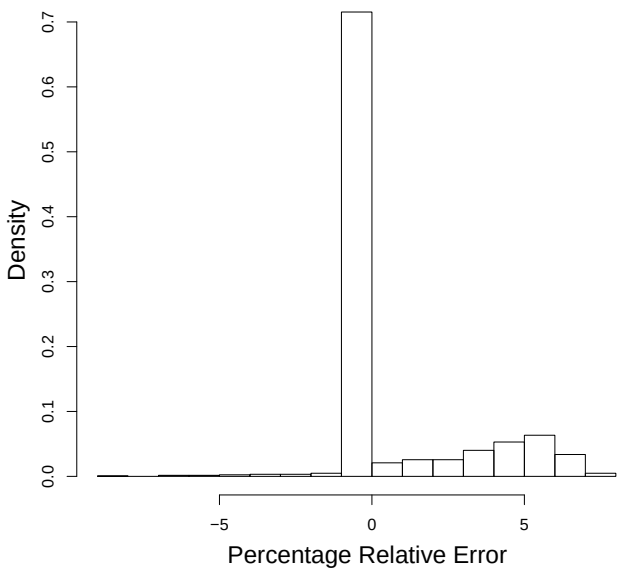


(b) Decision regions of the ADP algoritm with pure exploitation

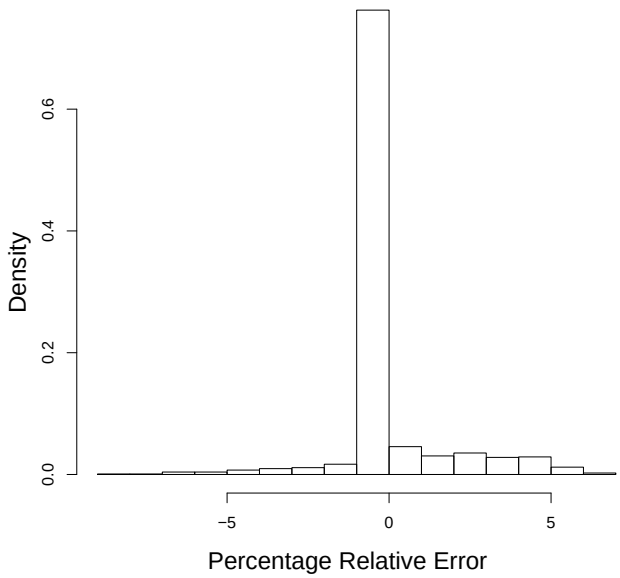


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.8: Decision Regions for Example 4: $a_{01} = a_{02} = 10, a_{12} = 16, a_{21} = 4, c = 1$

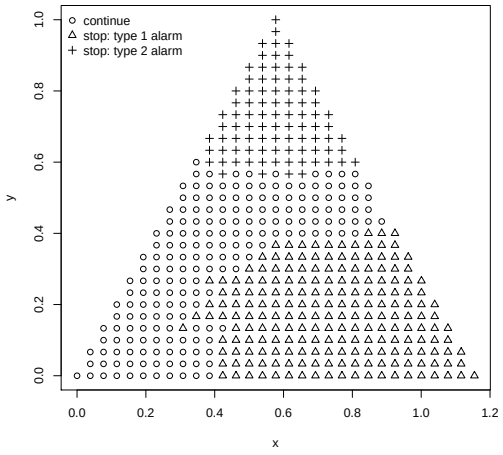


(a) Percentage relative error of the ADP algorithm with pure exploitation

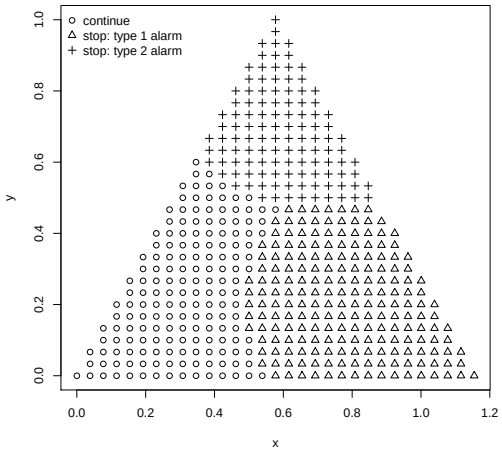


(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

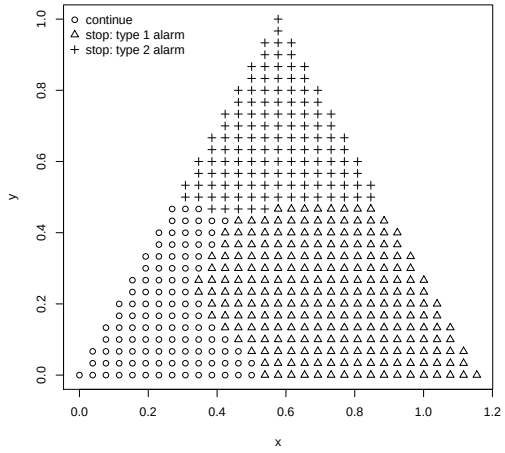
Figure 6.9: Histograms of Percentage Relative Error for Example 4: $a_{01} = a_{02} = 10, a_{12} = 16, a_{21} = 4, c = 1$



(a) Optimal decision regions

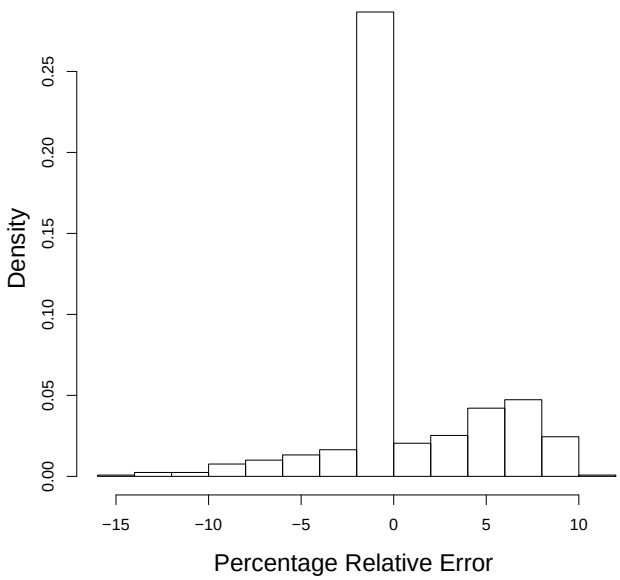


(b) Decision regions of the ADP algoritm with pure exploitation

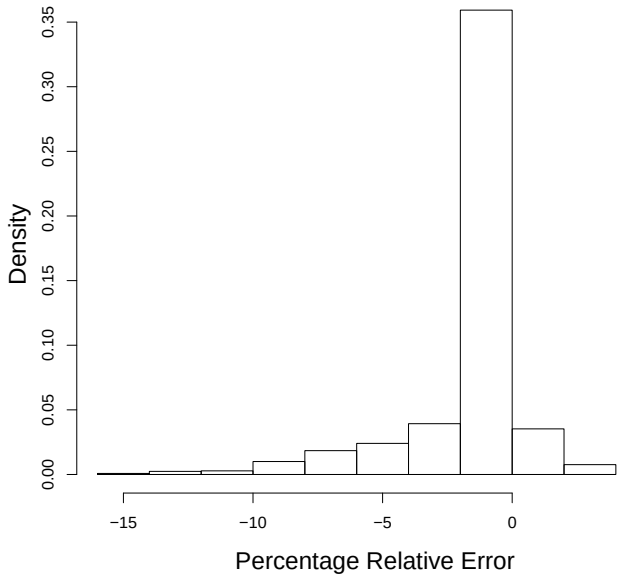


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.10: Decision Regions for Example 5: $a_{01} = 14, a_{02} = 20, a_{12} = a_{21} = 8, c = 1$

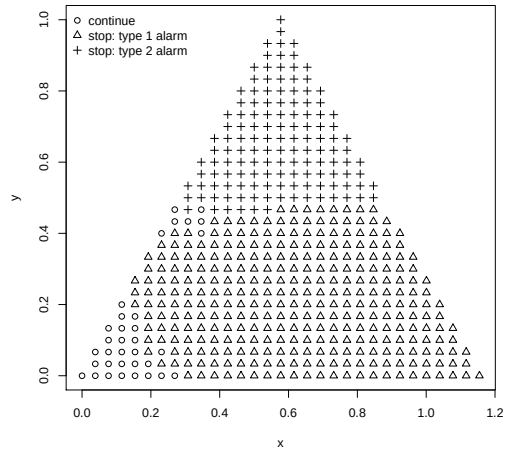


(a) Percentage relative error of the ADP algorithm with pure exploitation

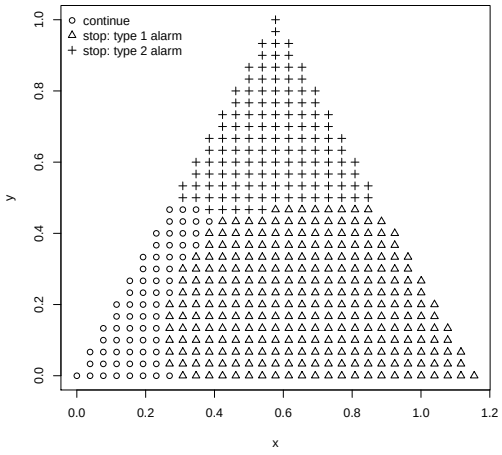


(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

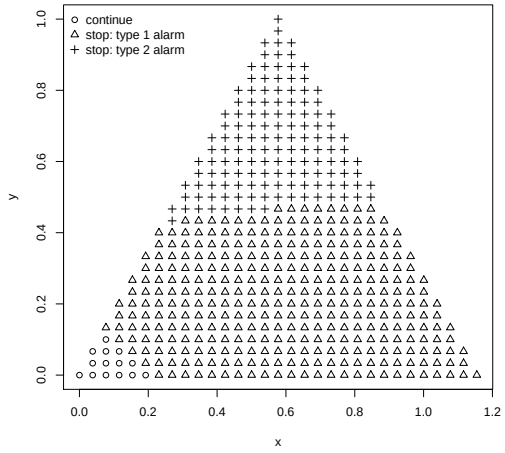
Figure 6.11: Histograms of Percentage Relative Error for Example 5: $a_{01} = 14, a_{02} = 20, a_{12} = a_{21} = 8, c = 1$



(a) Optimal decision regions

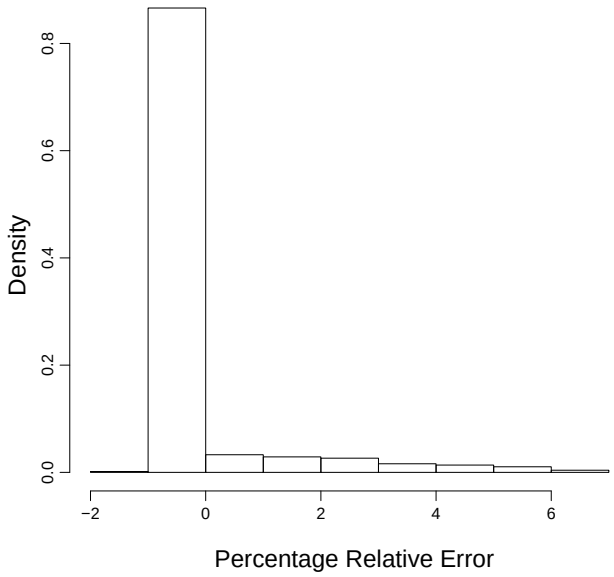


(b) Decision regions of the ADP algoritm with pure exploitation

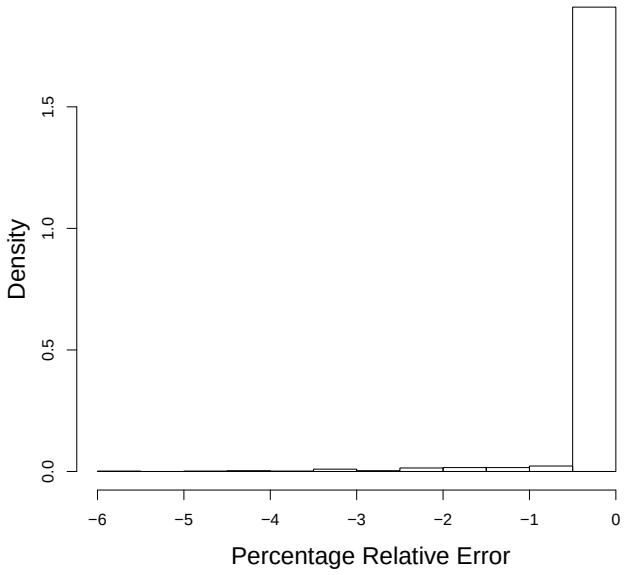


(c) Decision regions of the ADP algoritm with exploration rate $\rho = 0.5$

Figure 6.12: Decision Regions for Example 6: $a_{01} = 14, a_{02} = 20, a_{12} = a_{21} = 8, c = 2$

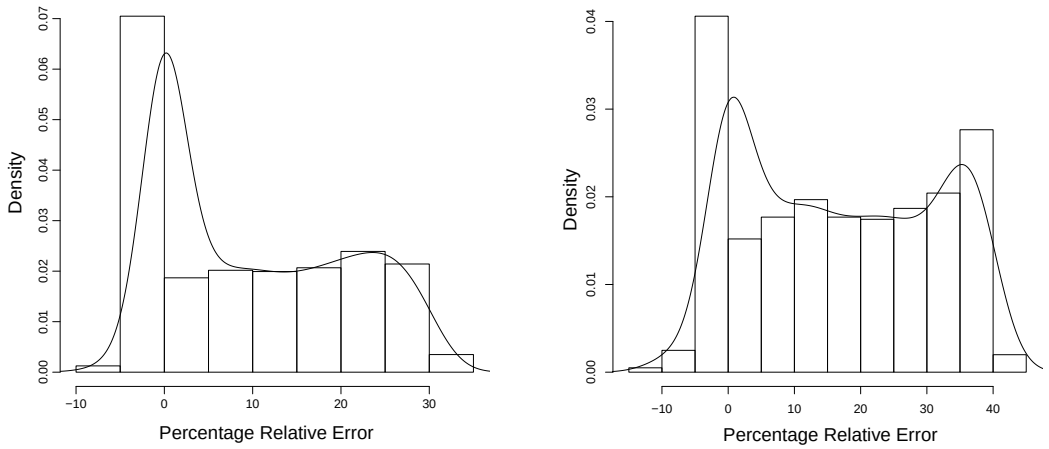


(a) Percentage relative error of the ADP algorithm with pure exploitation



(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

Figure 6.13: Histograms of Percentage Relative Error for Example 6: $a_{01} = 14, a_{02} = 20, a_{12} = a_{21} = 8, c = 2$



(a) Percentage relative error of the ADP algorithm with pure exploitation

(b) Percentage relative error of the ADP algorithm with exploration rate $\rho = 0.5$

Figure 6.14: An example with three alternative change distributions

Chapter 7

Conclusion

Sequential change diagnosis problem is the joint problem of change detection and sequential hypothesis. Although both change detection and sequential hypothesis testing problems have been studied extensively, there are few studies about sequential change diagnosis problem. Dayanık et al. [2] provide the complete optimal dynamic programming solution. However, due to curse of dimensionality, dynamic programming approach becomes computationally intractable for the problems with large number of alternative regimes after change. To overcome the curse of dimensionality, we have employed in this thesis the tools of approximate dynamic programming. The aim is to obtain an approximate continuation cost function. At any state, one can optimally decide between stopping and continuation to observe by comparing the continuation cost and stopping cost. We design an ADP algorithm which employs a basis function expansion of the continuation cost function. We represent the continuation cost function as an affine combination of radial basis functions. Linear mixed model is applied on radial basis functions. To evaluate designed ADP algorithm, we compare the solutions of the ADP algorithm to the optimal solutions in small problems. If the ADP reasonably approximates the optimal solution for the simpler problem, then we are more confident that the algorithm works fine [21] for the complex problem. One of the performance criteria is the similarity between approximate decision regions and

optimal decision regions. In Examples 1–6, the optimal and approximate decision regions are overlapping (see Figures 6.2, 6.4, 6.6, 6.8, 6.10and 6.12). We also compare the approximate continuation cost and the optimal continuation cost over a fine discritization of the state space. We compute the percentage relative error $\frac{C(\pi)-\bar{C}(\pi)}{C(\pi)} \times 100$ for every π in the discretized state space. The distribution of the percentage relative error indicates how well the algorithm approximates the continuation cost. We expect that the percentage relative error is accumulated around zero. Figures 6.3, 6.5, 6.7, 6.9, 6.11, 6.13 are the histograms of the percentage relative error for Examples 1–6. In Examples 1–6 the percentage relative error is distributed highly around zero. This shows that the algorithm reasonably approximates the optimal solution. We compare also pure exploitation strategy and mixed strategy with $\rho = 0.5$. Both pure exploitation and mixed strategies give reasonable approximation. Since the state space is not very large for $M = 2$, we manage to explore most of the states with pure exploitation strategy. Thus, there is little difference between decision regions in pure exploitation strategy and mixed strategy in Examples 1–6.

In Example 7, which is the special case in which $M = 3$ the percentage difference increases (see Figure 6.14). As M increases, the state space is getting larger and the number of unvisited states increases. Therefore, for complex problems with large M values, pure exploitation strategy tends to produce poor results. Simple heuristics such as using a constant exploration rate ρ may work well for small problems but has no value if the state space is large. Exploration rate ρ can be reduced after sufficient exploration. Determining how exploration rate should be decreased can be the subject of another research. Exploration vs. exploitation problem is a one of the unsolved problems in approximate dynamic programming. There is not a certain way to strike the balance between exploration and exploitation [3]. In future work, different exploration vs. exploitation strategies can be applied to the problems with large M values.

Bibliography

[1] S. Dayanik, W. Powell, and K. Yamazaki, "An asymptotically optimal strategy in sequential change detecton and identification applied to problems in biosurveillance," *3rd INFORMS Workshop on Data Mining and Health Informatic*, 2008.

[2] S. Dayanik, C. Goulding, and H. V. Poor, "Bayesian sequential change diagnosis," *Mathematics of Operations Research*, vol. 45, pp. 475–496, 2008.

[3] W. B. Powell, *Approximate Dynamic Programming: Solving the Curse of Dimensionality*. John Wiley and Sons, New York, NY, 2007.

[4] G. Lorden, "Procedures for reacting to a change in distribution," *Ann. Math. Statist*, vol. 42, pp. 1897–1908, 1971.

[5] A. N. Shiryaev, *Optimal Stopping Rules*. Springer-Verlag, 1978.

[6] A. Wald and J. Wolfowitz, "Bayes solutions of sequential decision problems," *Ann. Math. Statist*, vol. 21, pp. 82–99, 1950.

[7] D. Arrow, D. Blackwell, and M. A. Girshick, "Bayes and minimax solutions of sequential decision problems," *Econometrica*, vol. 17, pp. 213–244, 1949.

[8] M. Basselville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Prentice Hall Information and System Sciences Series, NJ, 1993.

[9] V. H. Poor and O. Hadjiliadis, *Quickest Detection*. Cambridge University Press, 2009.

[10] I. V. Nikiforov, "A generalized change detection problem," *IEEE Trans. Inform. Theory*, vol. 41, pp. 171–187, 1995.

[11] T. L. Lai, "Sequential multiple hypothesis testing and efficient fault detection-isolation in stochastic systems," *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 595–608, 2000.

[12] S. Dayanik and C. Goulding, "Detection and identification of an unobservable change in the distribution of a markov-modulated random sequence," *IEEE Transactions on Information Theory*, vol. 55, pp. 3323–3345, 2009.

[13] S. Dayank, W. Powell, and K. Yamazaki, "Asymptotically optimal bayesian sequential change detection and identification rules," *Annals of Operations Research. To appear in the special volume on Optimization under Uncertainty Costs, Risks, and Revenues in honor of Professor Cyrus Derman.*

[14] R. E. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, pp. 503–516, 1954.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Reinforcement Learning.* John Wiley and SonsThe MIT Press, Cambridge, Massachusetts, 1998.

[16] R. Sutton and A. Barto, *Reinforcement Learning.* John Wiley and SonsThe MIT Press, Cambridge, Massachusetts, 1998.

[17] R. E. Bellman and S. E. Dreyfus, "Functional approximations and dynamic programming," *Mathematical Tables and Other Aids to Computation*, vol. 13, pp. 247–251, 1959.

[18] D. Ruppert, M. P. Wand, and R. J. Carroll, *Semiparametric Regression.* Cambridge University Press, New York, NY, 2003.

[19] D. Nychka and N. Saltzman, "Design of air quality monitoring networks," *In D. Nychka, W.W. Piegorsch, and L. H. Cox (Eds.), Case Studies in Environmental Statistics (Lecture Notes in Statistics)*, vol. 132, p. 5176, 1998.

[20] M. E. Johnson, L. M. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of Statistical Planning and Inference*, vol. 26, pp. 131–148, 1990.

[21] W. Powell, "What you should know about approximate dynamic programming," *Naval Research Logistics*, vol. 56, p. 248, 2009.