Bandit feedback in Multiclass/Multilabel Classification and Multi-objective Optimization

Subtitle

By

HONGLIANG ZHONG

L'informatique Fondamental
ECOLE CENTRALE MARSEILLE

A dissertation submitted to Ecole Centrale Marseille in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of ED184.

30 JANVIER 2016

Abstract

Here goes the abstract

Dedication and acknowledgements

Here goes the dedication.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Part I

General Introduction

CHAPTER 1

INTRODUCTION

"Bandit problems embody in essential form a conflict evident in all human action: choosing actions which yield immediate reward vs. choosing actions whose benefit will come only later." – P. Whittle (1980)

Early in 1933, William R. Thompson described a method "Thompson Sampling" in article [64], for attempting to compare treatments which both have unknown effectiveness. Here, it is a focus on finding the balance between exploration and exploitation. This is the prototype of bandit problems.

In 1952, H. Robbins introduced a problem in [57], which worked in the area of sequential selection of experiments, and became the foundation of Bandit problem.

In 1957, Richard Bellman wrote the first book[13] at this subject, formulated the Multi-Armed Bandit problem as a class of dynamic program.

Related to H. Robbins's research, in 1979, Gittins and Jones published [40] to outline an allocation index for sequential experiment problems, and defined the stopping rule by the theorem.

From then on, more and more proofs of "Gittins Index" have been proposed, i.e. by Whittle [70], Weber [69], Tsitikis [66], ...

Nowadays, Further researches and applications of Bandit problem have been explored, for example, in Machine Learning by [15, 42, 63], Economics [5, 12, 61] , Cognitive Science [29], etc.

1.1 Modeling of Bandit Problems

Consider the Robbins's sequential decision experiment[57], there are several possible choices (finite arms or infinite arms). Generally, we should make a choice from the set of arms at each time. After observation, you will receive a boolean feedback **1** or **0** as reward. Whatever the feedback is, it is always the useful knowledge to make prediction in future from the past. We can even estimate the fixed but unknown distribution of arms by the reward of each time. The goal of bandit problems, is to maximize the received rewards from all steps, minimize the cumulative regret and find the optimal arm (the set of optimal arms).

The bandit problem can be defined by the number of trials, as the "finite horizon" or "infinite horizon". The former has a number of trials fixed and small, the latter's trials number usually unknown in advance, but has some probability at any trial will be the last; by the number of arms, it could be divided into "two arms", "K-armed" and "Many-armed" Bandits; by the environment, it is divided into "Stationary bandit", "Non-stationary bandit", "Adversary bandit" and "Contextual bandit".

In the next chapters, we will tell about each kind bandits, their performance, properties and caracteristics. Specially, we will emphasize on Chapter 3 and Chapter 4, to propose the novel solutions and algorithms. Here, we address a simple process in order to introduce the core thought of Bandit problems. Just like we have introduced , the target of bandit problems is to maximize the number of rewards over all steps. Generally, in the first few trials, pursuing the goal might involve trying different arms, getting some knowledge of which arms may be rewarding or which may not. Towards to the end, it will become increasingly important to pull the same arm repeatedly by the same reasonable strategy. Where the way to check out some uncertain arms is called "Exploration", and the ways to choose the arm (a set of arms) that is more certain by the reasonable strategy is called "Exploitation". The decisions are always depending on the potential knowledge of the optimal arms in the setting. If the setting is believed to have optimal arms, then others will be encouraged to try of arms. On the other hand, if the setting has a bad reputation for arm quality, a modest reward rate encourages repeat choice.

Performing well on bandit problems, requires to keep a balance between Exploration and Exploitation during decisions. In early steps, it makes sense to explore, to search for those with the highest reward rates. In later times, it makes sense to exploit those arms known to be good, maximizing the reward for each current decision. How to keep balance between Exploration and

Exploitation, (the details shown in the Chapter 2), it should be influenced by factors such as the distribution of reward rates, the total number of plays, etc.

We also attempt to construct the bandit models that capture individual differences and collective performance in solving bandit problems. Differences in how people solve the bandit problems that involve optimization and classification under interactive constraints, provide a window onto variance in cognitive abilities. Recently, take attention to some mechanism, a process of taking into account the collective ability of a group of individuals to solve a task, has discovered that a large group's aggregated answers to questions involving quantity estimation and general world knowledge are as good as, and often better than, the answer given by any of the individuals within the group (e.g.[62]). We are thus interested in whether this phenomenon also applies to sequential tasks such as bandit problems whether a collective decision-making through aggregating methods can achieve better outcomes than most of the individuals.

From the theorem view, we also care about how to find optimal solutions for bandit problems that make model evaluation and model comparisons more efficient. We dedicate next chapters of this dissertation to our effort on this problem.

1.2 Applications of Bandit Problems

The development of Bandit Problems is rapid, mainly due to its applications and prospects are very impressive.

Early in 1933, Dr. Thompson encountered a difficulty: to treat a same disease, there may be two or even more solutions, but their effectiveness are unknown. Actually, it is also the problem that Bandit problems need to solve, find the best choice on facing some unknown probability distributions. At that time, Thompson proposed the "Thompson Sampling"(details in Chapter 2), a prototype of Bandit problem.

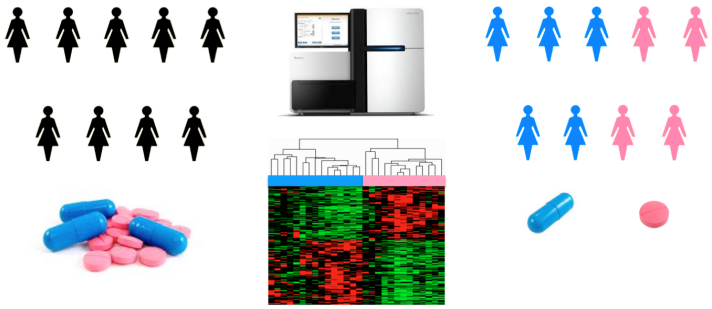

Figure 1.1: Clinical Trials

Nowadays, with the rising development of Internet, the Bandit problems become extremely popular. Such as personalized search, Internet advertising etc. For these applications, Operators will give each user a certain number of recommendations. sometimes, there are some user-friendly, but sometimes not. For traditional way, the supervised learning should get full feedback from

users, if there is no interested recommendation. But most users do not like give this full feedback. In this case, Bandit feedback is proposed.(Details will be shown in Chap 2)

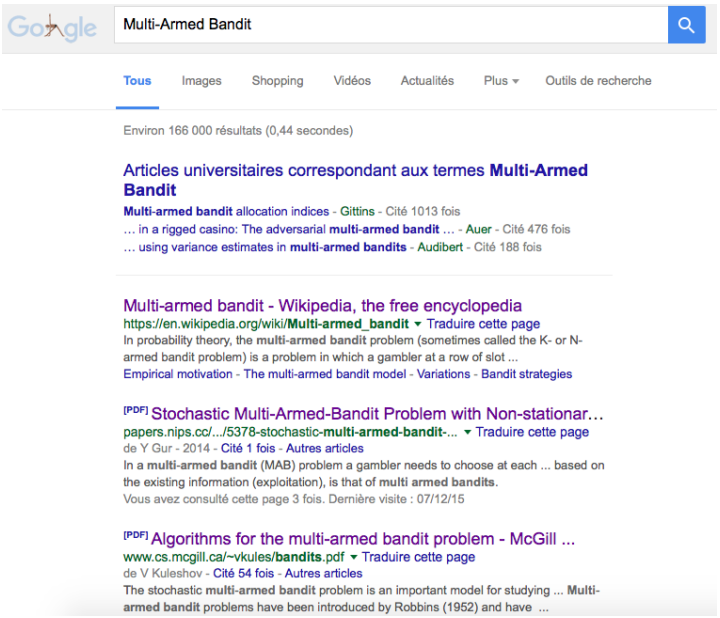

Figure 1.2: Web Search

Otherwise, Google proposed an analytic service based on the method of Multi-Armed Bandit(Chapter 2). It is an analytic system to compare and manage some online experiments. For each experimentation, maybe there are a variety of alternative solutions. In past, it is usually used A/B test to solve, but A/B test cost much and waste time. So google takes Bayes Theorem combining Multi-Armed Bandit, it could reduce much time and get higher accuracy than ever.

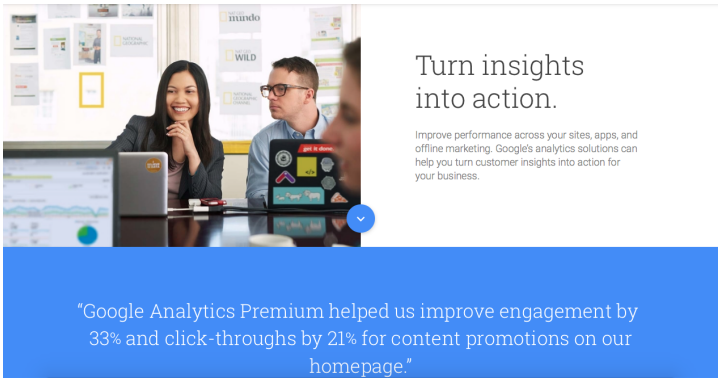

Figure 1.3: Google Analytic

Except Internet, Bandit problems also have very broad prospects in other domains. Such as in World War II, many countries had tried to solve multi-objective multi-tasking attack, this one could be considered as Multi-Objective Multi-Armed Bandit problem (shown in Chapter 4).

Otherwise, queueing and scheduling([33, 68]), stock in non-stationary([36]) etc, all of them will be the direction to be committed to develop for Bandit problems.

1.3 The structure of thesis

This dissertation consists of 5 Chapters followed the discussion of Bandits Problem. This first Chapter, proposed the background of Bandit problems.

Chapter 2, is about some important strategies of Bandit problem, how to trade off the balance between Exploration/Exploitation. In this chapter, we introduce some methods very useful and effective to keep balance for the tradeoff. Furthermore, to analyze their advantage and disadvantage with principal theorem and empirical experiments. The classical Bandit problem, Multi-Armed Bandit problem, will be introduced. Recently, there are many researches working in this setting. We concentrate on its environment, lower bound, gittins index and best arm identification etc.

Chapter 3, it's the classification problem under Bandit Feedback. Here, we proposed several algorithms in the-state-of-art to solve the problems in multiclass or multilabel classification. Compared those algorithms, we will introduce the algorithms who related or modified from those classification algorithms. At last, compare their effectiveness or regret bound by empirical experiments based on those common data sets.

Chapter 4, it is the extension of Multi-Armed Bandit that each arm has multi-objective. So many domains are met this problem, i.e. supply chain, some decision system, Big data platform and so on. In this chapter, we will introduce some traditional way to solve this problem, and our contribution, a novel method who works more effectively and more precisely.

Chapter 5, also the last chapter. We will make the conclusion of all chapters in this part.

Part II

State of the art

CHAPTER 2

MULTI-ARMED BANDIT

Bandit problems just as we expounded in Chapter 1, was formally proposed in 1952 by H.Robbins. Generally, this issue could be simplified such a model as below:

A gambling machine with several arms, each of which has an unknown, possibly different distribution of payoffs. The gambler has no idea about which arm with most reward. Considering a sequential decision experiment, the arm k from all arms in an observation being taken from the i^{th} experiment, and the gambler receives a numerical value of this observation as a reward. The observations maybe give him some information useful in the future choices. However, this observation to obtain information is also to gain rewards. So, the problem for the gambler is how to strike a balance between gaining rewards and obtaining information.

For example, it is not good to always pull the arm that has performed best in the past, because it may have been that you were just unlucky with the best arm. If you have many trials to go and it only takes a few trials to clarify the matter, you can stand to improve your average gain greatly with only a small investment. Typically in these problems, there is a period of gaining information, following by a period of narrowing down the arms, followed by a period of "profit taking", playing the arm you feel to be the best.

That model is also called “Multi-Armed Bandit” (MAB), which is a classical Bandit problem. In this chapter, we focus on describing MAB to understand Bandit problems. The purposes of this chapter is to propose some necessary notions, present some effective strategies, define and analyze some specific issues.

2.1 The environment of Multi-Armed Bandit

The complexity of Multi-Armed Bandit problems, is not only from the decision-making between Exploration and Exploitation, but also by the wide range of its environments. Each environment to the Multi-Armed Bandit has a different framework. In this section, we address to present the characteristics and performance of Multi-Armed Bandit in each environment, and take the K-Armed Bandit as the typical case.

2.1.1 Stationary Bandit

Stationary Bandit[57] is the most classical and common bandit problem. The gambler should pick up an arm from the arm setting, where all arms give him a reward for the response. The reward obeys a fixed probability distribution and independent to others arms. The K-Armed Bandit in stationary environment, can be defined as following setting. The gambler faces to the slot machine with K arms. Where each arm k from the arm set $\mathcal{K} = \{1, \ldots, K\}$ is characterized by a distribution ν_k with its mean reward μ_k and a variance σ_k^2, all of these parameters are no changeable. At each round $t \geqslant 1$, the gambler selects an arm k_t and receives a sample drawn from ν_{k_t} independently of the past. Here, the goal of the gambler is to maximize rewards (minimize regret) or to identify the best arm from all arms by estimating mean rewards after T pulls.

Under T times pull and observation, gambler samples from each arm will be denoted as $T_k = \sum_{t=1}^{T} \mathbb{1}(k = k_t)$. Finally, gambler could estimate the mean of each arm by $\hat{\mu}_k = \frac{1}{T_k} \sum_{t=1}^{T_k} X_{k_t,t}$, where $X_{k_t,t}$ denotes the sample received when we pull arm k for the t^{th} time. After T times observation, we find the optimal arm k^*:

$$k^* = \underset{k \in \mathcal{K}}{\operatorname{argmax}}\ \hat{\mu}_k \text{ and } \quad \mu^* = \max_{k \in \mathcal{K}} \hat{\mu}_k \tag{2.1}$$

Simple Regret In the sequel, $\Delta_k = \mu^* - \hat{\mu}_k$ is the gap between the maximal expected reward and the k^{th} arm of all arms $\mathcal{K}$. And the minimal gap can be noted by $\Delta = \min_{k:\Delta_k > 0} \Delta_k$. So, the simple regret at round T equals the regret on a one-shot instance for the chosen arm k_T, that is,

$$r_T = \mu^* - \hat{\mu}_{k_T} = \Delta_{k_T}. \tag{2.2}$$

Regret In the stochastic framework, we define the cumulative regret as:

$$R_T = \sum_{t=1}^{T} (\mu^* - \hat{\mu}_{k_t}) \tag{2.3}$$

2.1.2 Adversary Bandit

Adversary environment[9], a branch of non-stationary environment, is a challenging problem of the Multi-Armed Bandit. In this environment, the rewards for each step are selected by an

adversary rather than the stationary environment where rewards being picked from a fixed distribution. Any method to solve the MAB in adversary environment should recognize that the information between the gambler and the adversary is asymmetry. Similarly to the stationary environment, we define $\mathscr{K} = \{1,\dots,K\}$ be a set of arms, and $\mathscr{T} = \{1,2,\dots,T\}$ denote the sequence of decision epochs by gambler. To compare between the stationary environment and the adversary environment, the difference is the reward distribution of arms. In adversarial environment, the reward distribution ν is changeable at each epoch t under adversary's control instead of the constant distribution in stationary. From another point of view, adversarial bandit can be seen as a competition between the gambler and an omniscient adversary.

This issue can be transformed into three step at the iteration t:

- the adversary choose the reward distributions $\mu_{\mathbf{t}} = (\mu_{\mathbf{1,t}},\dots,\mu_{\mathbf{K,t}}) \in [\mathbf{0},\mathbf{1}]^{\mathbf{K}}$
- the gambler picks one arm k_t with no knowledge of the adversary's choice
- the rewards are assigned $X_{k,t} \sim \mu_{k,t}$

Regret who measures the performance of the gambler compared to the performance of the best arm:

$$R_T = \max_{k=1,\dots,K} \sum_{t=1}^{T} \mu_{k,t} - \sum_{t=1}^{T} X_{k_t,t}.$$

2.1.3 Contextual Bandit

Contextual Bandit [4, 53], a natural extension of the Multi-Armed Bandit problem, is obtained by associating side information with each arm. Based on this side information, or context, some applications could be modeled naturally as Multi-Armed Bandit problem with context information, so it also named as Bandit with side information. As it is closely related to work in machine learning on supervised learning and reinforcement learning. In order to facilitate the presentation in Chapter 3, the definitions of Contextual Bandit, will be referred to some notations in supervised learning.

For contextual bandit, there is a distribution $\mathbb{P}$ over $(x, r_1, r_2, \dots, r_K)$. On each round $t \in \mathscr{T} = \{1,2,\dots\}$, the gambler receives a sample $(x_t, r_{t,1}, r_{t,2}, \dots, r_{t,K})$ drawn from $\mathbb{P}$ and makes decision $\hat{y}_t \in \mathscr{Y}$ by observing the set of arms $\mathscr{Y} = \{1,\dots,K\}$ with side information a feature vector $\mathbf{x}_{t,y_t} \in \mathscr{X}$. Where y_t is the optimal arm for x_t, but gambler do not know this. With the chosen arm $\hat{y}_t$, gambler receives a reward $r_{(x_t,\hat{y}_t)}$. We should emphasize that the reward is only observed for the arm chosen .

After $\mathscr{T}$ sequence, all rewards for gambler is defined as $\sum_{t=1}^{T} r_{(x_t,\hat{y}_t)}$. Similarly referring to the stationary setting, we define the regret by the notations above:

$$R(T) = \mathbb{E}[\sum_{t=1}^{T} r_{(x_t,y_t)}] - \mathbb{E}[\sum_{t=1}^{T} r_{(x_t,\hat{y}_t)}].$$

Contextual bandits naturally arise in many applications. For example, online recommendation systems, advertisement push, personalized search etc.

2.1.4 Linear Bandit

Linear Bandit problem[1, 20] is also a sequential decision-making problem as same as other bandit problem, where in each step, the gambler has to choose an arm from the arms set. As a response, the gambler receives a stochastic reward, expected value of which is an unknown linear function of the chosen arm.

Here, we present formally the process of Linear Bandit problem. On round t, gambler is given the arms set $\mathscr{K} \subseteq \mathbb{R}^d$. From which he should select an arm $k_t \in \mathbb{R}^d$. Then, the gambler observes a reward $r_t =< k_t, \theta > +\eta_t$ where $\theta \in \mathbb{R}^d$ is an unknown parameter and η_t is a random noise satisfying the condition $\mathbb{E}[\eta_t | k_{1:t}, \eta_{1:t-1}] = 0$.

As other bandit problems, the goal of this problem is to maximize the cumulative reward $R = \sum_{t=1}^{T} < k_t, \theta >$ over all round T. Obviously, the gambler would choose the optimal arm $k^* = \underset{k \in \mathscr{K}}{\operatorname{argmax}} < k, \theta >$ with the knowledge of θ. As a result, the regret of linear bandit problem, can be formally denoted as

$$R = \left(\sum_{t=1}^{T} < k_t^*, \theta > \right) - \left(\sum_{t=1}^{T} < k_t, \theta > \right) = \sum_{t=1}^{T} < k_t^* - k_t, \theta >$$

.

2.2 Gittins index

Multi-Armed Bandit is concerned with sequential decision. The definition can be found at the head of this chapter. In this section, we focus on his optimal decision process. Someone treat MAB problem as a Markov Decision Process (MDP). However, there is no approach can scale well it on considering the curse of dimensionality by the number of bandit process. In that situation, Gittins and Jones [38] show that the problem can be reduced to solve K 1-dimensional problems from the K-dimensional Markov Decision Process with the state space $\prod_{k=1}^{K} X(k)$, where $X(k) = (x(k,1), x(k,2), \dots)$ present the state of arm k. So that, K-armed bandit returns denoted by

$$\begin{aligned} \text{arm } 1 &: x(1,1), x(1,2), \dots, x(1,t), \dots \\ \text{arm } 2 &: x(2,1), x(2,2), \dots, x(2,t), \dots \\ &\dots \\ \text{arm } K &: x(K,1), x(K,2), \dots, x(K,t), \dots \end{aligned}$$

For each arm k, to compute

$$\mathscr{G}_k(X(k)) = \sup_{\tau > 0} \frac{\mathbb{E}[\sum_{t=0}^{\tau-1} \beta^t r_k(x(k,t)) | x(k,0) = x(k)]}{\mathbb{E}[\sum_{t=0}^{\tau-1} \beta^t | x(k,0) = x(k)]} \tag{2.4}$$

where τ is a stopping time constrained and $\mathcal{G}_k$ is the value of the Gittins index for arm k. The stopping time τ represents the first time at which the index for this arm may be optimal no more. For its index neither greater than the initial value. By the way, the decision rule is then to simple arm k_t, which can be computed by the format $k_t = \underset{k \in \mathcal{K}}{\operatorname{argmax}}\, \mathcal{G}_k(X(k))$.

Theorem 2.1. *Gittins Index Theorem The problem posed by a simple family of alternative bandit processes, as setup above, is solved by always continuing the process having the greatest* ***Gittins Index****, which is defined as Function 2.4.*

To this theorem, there are various proofs have been given, the original proof is proposed by Gittins and Jones[38], and a later proof by Gittins [39] relied on an interchange argument. Further simplified by [66, 67]. More details about the proofs can be referred in their papers.

Gittins Index characterized the optimal strategy as Theorem 2.1, an equivalent interpretation of the Gittins Index strategy is the following:

- Select the arm with the highest Gittins Index $\mathcal{G}_k(X(k))$ and play it until its optimal stopping time and repeat.

Thus, an alternative way to implement the optimal way is to compute the value of Gittins index $\mathcal{G}_k(X(k))$ and the corresponding stopping time τ for the current state $x(k,t)$.

Scott [59] points out two further concerns that the Gittinx Index strategy is only optimal for the process in which arms are independent and can be far from optimal when this is not the case.

2.3 The strategy of trade-off

The synopsis of Bandit problems have been introduced in Section 2.1. From this synopsis, we get the difficult point of Bandit problem is to keep a balance between Exploration and Exploitation. So that, it becomes very important to trade off between to exploit from the past knowledge to focus on the arms that seems to yield the highest rewards and to explore further the other arms to identify that better which may be the actual global optimal.

The easiest way to solve this problem, is to select randomly. However, this way mainly relies on luck. So it is not reliable. Another simple way is called “Naive Sampling” approach. At early times, it samples each arm by same number, and then measure the results on comparing these samples. After that, an optimal empirical solution has been shown. But this approach also has a problem, that is, if the initial number of samples is too small, the confidence will be low; if the initial number are huge enough, will waste the cost. Therefore, in this section, we will introduce some effective strategies to solve bandit problems.

2.3.1 Thompson Sampling

Thompson Sampling [64], a randomized algorithm based on Bayesian ideas, is one of oldest heuristic principle for Bandit problems. Recently, it has been considered having better empirical performance compared to the state-of-the-art methods after several studies demonstrated [3, 4, 23].

The origins of Thompson Sampling has been introduced in Chapter 1, is from the procedure's inventor William R. Thompson. Thompson was interested in the general problem of research planning. He was concerned with being able to utilize a small numbers of observations in order to steer actions taken before more data could be collected. This was in the context of a perceived objection to argument based on small numbers of observations at the time. Thompson posed his problem in terms of deciding between two treatments in a clinical trial. One of the treatments would be administered to a patient in the trials population and the effect could be observed. These observations could then be incorporated into the decision-making process to improve the odds of the most effective treatment being administered to further patients.

For Bandit problems, this randomized strategy will choose an arm with some probability which matches the probability that the arm is in fact the optimal arm, by giving all past observations of all arm pulls. More reasonable for this randomized chosen is to define the probability that an arm is the best is a Bayesian estimate.

Being more precise, let θ be a parameter vector on behavior of a bandit problem. The probability that an arm k is optimal,

$$P(K = k^*) = \int_\theta \mathbb{1}(K = k^*|\theta)P(\theta)\mathrm{d}\theta. \tag{2.5}$$

An arm is thus pulled with the probability $P(K = k^*)$. This Sampling way could be viewed as forming a decision based on one-step Monte-Carlo Sample by estimating the probability of an arm being the best.

The integral above may not have a closed form solution. The integral may be approximated by quadrature. However, this is undesirable as the running cost of each decision will depend on the difficulty of the multi-armed bandit problem. The insight used in Thompson Sampling is that instead we can sample from a distribution defined by $P(K = k^*)$ by simply first sampling a candidate θ from the distribution $P(\theta)$. Given a candidate θ we can then just pull the arm that is best given this candidate ($\mathbb{1}(K = k^*|\theta)$). $P(\theta)$ is initially specified as a prior and then later inferred using Bayes rule as rewards from arm pulls are observed . The general Thompson Sampling algorithm for a K-armed bandit is therefore described as following

Algorithm 2.1 (Thompson Sampling)**.**

Initialise $P_{(\mu_1,\ldots,\mu_K)}$*, the prior belief of the mean payoffs of arms* $1,\ldots,K$.

Let H_t *be the history of action, reward pairs* (r_τ, k_τ) *for* $1 \leqslant \tau \leqslant t$, $H_1 = \{\}$

for *each round t = 1,2,. . . , T* ***do***

Sample $\theta_i,\ldots,\theta_K \sim P(\mu_1,\ldots,\mu_K|H_t)$.

Pull arm $k_t = argmax_{k \in \{1,\dots,K\}} \theta_k$
Receive reward $\mathbb{1}_{r(k_t)=1}$
Let $H_{t+1} = H_t \cup (k_t, \mathbb{1}_{r(k_t)=1})$.
end for

Optimism in Thompson Sampling There is a question, what is the tradeoff between exploration and exploitation for Thompson Sampling. May [53] tried to separate the two aspects of this algorithm. To do this, he defined the exploitative value of an arm to be the expected payoff of an arm conditioned on the rewards. The estimated value of an arm could be seen as the value of the sampling drawn from the posterioi distribution for the arm. With these, the exploratory value of an arm could then by found by subtracting the exploitative value from the estimated sample value. They observed that this exploratory value could sometimes be negative and so there would be no value from an exploration point of view to pull the arm. The exploratory value of an arm is only negative when the sample estimate drawn from the posterior is less than the exploitative value of the arm.

In Thompson Sampling, samples are drawn from the posterior distribution of each arm, that is $\theta_k(t) \sim P_{(\mu_k)}$. Instead, Optimistic Thompson Sampling drawn samples such that $\theta_k(t) = max(\mathbb{E}[\mu_k], s_k(t))$ where $s_k(t) \sim P_{(\mu_k)}$. In other words if a sample from the posterior distributions is less than the mean of the distribution, then we take the sample to be then mean. This ensures that the exploratory value of an action is always positive. In Appendix Algorithm A.1 more formally presents the algorithm specifically for the Bernoulli bandit. May[53] observed empirically that Optimistic Thompson Sampling performed better than standard Thompson Sampling.(See in Appendix A.1)

2.3.2 Boltzmann Exploration (Softmax)

This section, is about Boltzmann Exploration (also named Softmax), which is based on the axiom of choice [51] and pick each arm with a probability that is proportional to its average behavior. Arms with greater empirical means could be therefore picked with higher probability. Softmax selects arms by using a Boltzmann distribution. Given initial empirical means $\hat{\mu}_1(0), \dots, \hat{\mu}_K(0)$,

$$p_k(t+1) = \frac{\exp \hat{\mu}_k(t)/\tau}{\sum_{i=1}^{K} \exp \hat{\mu}_i(t)/\tau}, \text{where } k \in \{1,\dots,K\} \tag{2.6}$$

Softmax may depend on the task and on human factors by the only parameter τ. Where τ is a temperature parameter, controlling the randomness of the choice. When $\tau = 0$, Boltzmann Exploration acts like pure greedy. On contrast, τ tends to infinity, the algorithms picks arms uniformly at random. Most people find it easier to set the τ requires knowledge of the likely action values and of powers of e.

Algorithm 2.2 (SoftMax)**.**
Parameter: real number $\tau > 0$

Initialization: Set $\hat{\mu}_k(0) = 0$ *for* $\forall k \in [1,\ldots,K]$
for *each round t = 1,2,..., T* ***do***
Sample arms i according to the distribution P(t), where

$$P_k(t) = \frac{\exp\left(\hat{\mu}_k(t-1)/\tau\right)}{\sum_{i=1}^{K}\exp\left(\hat{\mu}_i(t-1)/\tau\right)}$$

Receive the reward r_{k_t}*, here* k_t *is the sampled arm for instant t*
Let $\hat{\mu}_k(t) = \sum_t r_{k,t}/\sum_t \mathbb{1}_{k=k_t,t}$
end for

The Softmax strategy could be modified in the same way as the ϵ-greedy strategy in Section 2.3.4 into decreasing Softmax where the temperature decreases with the number of rounds played. The decreasing Softmax is identical to the Softmax but with a temperature $\tau_t = \tau_0/t$ that depends on the index t of the current round. The choice of the value of τ_0 is left to the user. The decreasing Softmax is analyzed by Cesa-Bianchi [22] with the Softmix algorithm.

A more complicated variant of the Softmax algorithm, the EXP3 "(see in Appendix A.2)exponential weight algorithm for Exploration/Exploitation" is introduced in [10]. The probability of choosing the arm k at the round of index t is defined by

$$P_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{i=1}^{K} w_i(t)} + \frac{\gamma}{K} \tag{2.7}$$

where $w_i(t+1) = w_i(t)exp\left(\gamma\frac{r_i(t)}{P_i(t)K}\right)$, if the arm i has been pulled at time t with $r_i(t)$ being the observed reward, $w_i(t+1) = w_i(t)$ otherwise. The choice of the value of the parameter $\gamma \in (0,1]$ is left to the user. The main idea is to divide the actual gain $r_i(t)$ by the probability $P_i(t)$ that the action was chosen. For a modified version of EXP3, with γ decreasing over time, it is shown by [9], that a regret of $O(\sqrt{KT\log K})$ is achieved.

2.3.3 Upper Confidence Bound

Upper Confidence Bound (UCB) was proposed by Lai [47], to deal with the Exploration and Exploitation dilemma in the Multi-Armed Bandit problem by using Upper Confidence Values. Most strategy for trading off Exploration and Exploitation, they have one weakness: they do not keep track of how much they know about any of options available to them. They pay much more attention only to how much reward they got. This means that they will under-explore options whose initial experiences were not rewarding, even though they don't have enough data to be confident about those options. But UCB can do better that pays attention to not only what it knows, but also how much it knows.

For example, in stochastic Multi-Armed Bandit problem, gambler has to choose in trials $t \in \{1,2,\ldots,T\}$ an arm from a given set of arms $\mathscr{K} = \{1,\ldots,K\}$. In each trial t the gambler obtains

random reward $r_{k,t} \in \{0,1\}$ for choosing arm k. It is assumed that for each arm k the random reward $r_{k,t}$ is independent and identically distributed random variables with mean μ_k which is unknown. Further, it is assumed that the rewards $r_{k,t}$ and $r_{k',t'}$ for distinct arms k,k' are independent for all $k \neq k' \in \mathscr{K}$ and all $t,t' \in \mathscr{T}$.

The gambler's aim is to compete with the arm giving highest mean reward $\mu^* := max_{k \in \mathscr{K}} \mu_k$. The arm with the best estimate $\hat{\mu}^*$ so far serves as a creteria, and other arms are played only if the upper bound of a suitable confidence interval is at least $\hat{\mu}^*$. That way, within T trials each suboptimal arm can be shown to be played at most $\left(\frac{1}{D_{KL}} + o(1)\right) \log T$ times in expectation, where D_{KL} measures the distance between the reward distributions of the optimal and the suboptimal arm by the Kellback-Leibler divergence, and $o(1) \to 0$ as $T \to \infty$. This bound was also shown to be asymptotically optimal[47].

Auer [7] introduced the simple, yet efficient UCB algorithm, that is also based on the ideas of Lai's. After playing each arm once for initialization, UCB chooses at trial t the arm k that maximizes

$$\hat{\mu}_k + \sqrt{\frac{2 \log t}{n_k}} \tag{2.8}$$

, where $\hat{\mu}_k$ is the average reward obtained from arm k, and n_k is the number of times arm k has been played up to trial t. The value in Equation 2.8 can be interpreted as the Upper Bound of a confidence interval, so that the true mean reward of each arm k with high probability is below this upper confidence bound.

In particular, the upper confidence value of the optimal arm will be higher than the true optimal mean reward μ^* with high probability. Consequently, as soon as a suboptimal arm k has been played sufficiently often so that the length of the confidence interval $\sqrt{\frac{2 \log t}{n_k}}$ is small enough to guarantee that

$$\hat{\mu}_k + \sqrt{\frac{2 \log t}{n_k}} < \mu^*$$

, arm k will not be played anymore with high probability. As it also holds that with high probability

$$\hat{\mu}_k < \mu_k + \sqrt{\frac{2 \log t}{n_k}}$$

, arm k is not played as soon as

$$2\sqrt{\frac{2 \log t}{n_k}} < \mu^* - \mu_k$$

, that is, as soon as arm k has been played

$$\left\lceil \frac{8 \log t}{(r^* - r_k)^2} \right\rceil$$

times. This informal argument can be made stringent to show that each suboptimal arm k in expectation will not be played more often than $\frac{\log T}{\Delta_k^2}$ times within T trials, where $\Delta_k := \mu^* - \mu_k$ is the distance between the optimal mean reward and μ_k.

Algorithm 2.3 (Improved UCB algorithm)**.**

Set arms $\mathscr{K} = \{1,\ldots,K\}$*, all playing times* $\mathscr{T} = 1,2,\ldots,T$

Initialization: Set $\tilde{\Delta}_0 := 1$*, and* $B_0 := \mathscr{K}$*.*

for *each round* $t = 1,2,\ldots,\lfloor \frac{1}{2}\log\frac{T}{e} \rfloor$ ***do***

Select arm: if $|B_m| > 1$*, choose each arm in* B_m *until the total number of times it has been chosen is* $n_m := \left\lceil \frac{2\log T\tilde{\Delta}_m^2}{\tilde{\Delta}_m^2} \right\rceil$*. Otherwise choose the single* B_m *until step* T *is reached.*

Eliminate arm: Delete all arms i *from* B_m *for which*

$$\left\{ \hat{\mu}_i + \sqrt{\frac{\log T\tilde{\Delta}_m^2}{2n_m}} \right\} < max_{j\in B} \left\{ \hat{\mu}_j - \sqrt{\frac{\log T\tilde{\Delta}_m^2}{2n_m}} \right\}$$

in order to obtain B_{m+1}*. Here* $\hat{\mu}_j$ *is the average reward obtained from arm* j*.*

Reset $\tilde{\Delta}_m$*: Set* $\tilde{\Delta}_{m+1} = \frac{\tilde{\Delta}_m}{2}$

end for

In [11], Auer proposed an improved UCB algorithm (shown in Algorithm 2.3). In this improved model, gambler can access to the values Δ_k, one could directly modify the confidence intervals of UCB as given [2] to $\sqrt{\frac{2\log t\Delta_k^2}{n_k}}$, and the proof of the claimed regret bound would be straightforward.

However, since the Δ_k is unknown to the learner, the modified algorithm shown in Algorithm 2.3 guesses the values Δ_k by a value $\tilde{\Delta}$, which is initialized to 1 and halved each time the confidence intervals become shorter than $\tilde{\Delta}$. Note that compared to the original UCB algorithm the confidence intervals are shorter, in particular for arms with high estimated reward. Unlike the original UCB algorithm, our modification eliminates arms that perform bad. As the analysis will show, each suboptimal arm is eliminated as soon as $\tilde{\Delta} < \frac{\Delta_k}{2}$, provided that the confidence intervals hold. Similar arm elimination algorithms were already proposed in [34].

2.3.4 Epsilon-Greedy

In this section, we are going to introduce a simple algorithm for trading off exploration and exploitation. This strategy is called ϵ-Greedy[63] (shown in Appendix 2.4). In computer science, a greedy algorithm is an algorithm that always takes whatever action seems best at the present moment, even when that decision might lead to bad long term consequences. The ϵ-greedy algorithm is almost a greedy algorithm because it generally exploits the best available option, but also have some chance to explore the other available options.

Let's be more concrete to the mechanism of ϵ-greedy algorithm. It works by randomly oscillating between the purely randomized experimentation and instinct to maximize profits. The ϵ-greedy is one of the easiest bandit algorithms to understand because it tries to be fair to the two opposite goals of exploration and exploitation by using a mechanism (see the Figure 2.1). Take a simple example to understand easily: it is just like flipping a coin. If you flip a coin and it comes up heads, you should explore for a moment. But if the coin comes up tails, you should exploit.

The formal process as follows,

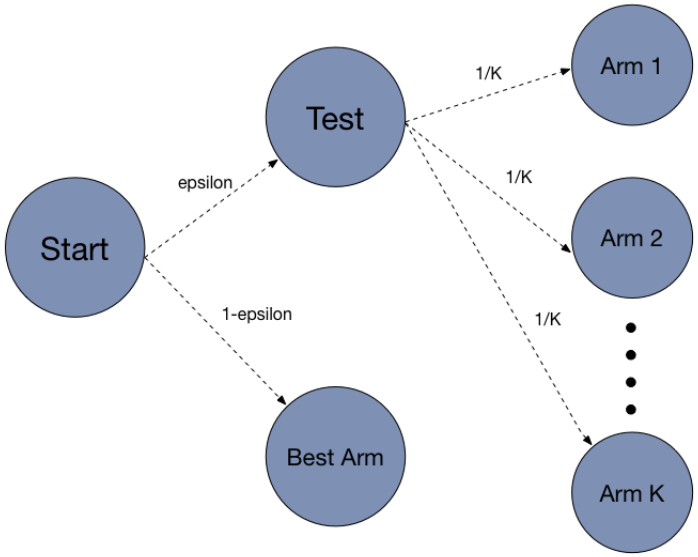

Figure 2.1: the mechanism of epsilon-greedy

- Firstly, pick a parameter $\epsilon \in [0,1]$,
- Then, at each step greedily play the object with highest empirical mean reward with probability $1-\epsilon$ and play a random arm with probability ϵ.
- Receives the reward from the arm chosen
- Finally, update the empirical means of each arm.

Algorithm 2.4 (ϵ-Greedy)**.**

Initialise $P_{(\mu_1,\ldots,\mu_K)}$*, the prior belif of the mean payoffs of arms* $1,\ldots,K$.

for *each round t = 1,2,. . . , T* ***do***

$$\textit{Pull arm } k_t = \begin{cases} \textit{argmax}_{k\in\{1,\ldots,K\}} P_{(\mu_k)} & \textit{with probability } \epsilon \\ \textit{select randomly} & \textit{with probability } 1-\epsilon \end{cases}$$

Receive reward $r_{(k_t)}$

Update μ_{k_t} *by the reward* $r_{(k_t)}$.

end for

Auer [8] has proven that, if ϵ is allowed to be a certain function ϵ_t following the current time step t, namely $\epsilon_t = K/(d^2 t)$, then the regret grows logarithmically like $(K \log T/d^2)$, provided d less than the number of objects with minimum regret . While this bound has a suboptimal dependence on d. In Auer's [8] same paper, it show that this algorithm performs well in practice, but the performance degrades quickly if d is not chosen as a tight lower bound.

Compared to other more complex methods, ϵ-greedy is often hard to beat and reported to be often the method of the first choice as stated . In practice, however, a drawback of ϵ-greedy is that

it is unclear which setting of ϵ leads to good results for a given learning problem. For this reason, the experimenter has to rigorously hand tune ϵ for obtaining good results, which can be a very time-consuming task in practice depending on the complexity of the target application.

One method that aims at overcoming the above mentioned limitation of ϵ-greedy is "Value-Difference Based Exploration"(VDBE)[65]. In contrast to pure ϵ-greedy, VDBE adapts a state-dependent exploration-probability. The basic idea of VDBE is to extend the ϵ-greedy method by controlling a state-dependent exploration probability, $\epsilon(s)$, in dependence of the value-function error instead of manual tuning. The desired behavior is to have the agent more explorative in situations when the knowledge about the environment is uncertain, i.e. at the beginning of the learning process, which is recognized as large changes in the value function. On the other hand, the exploration rate should be reduced as the agent's knowledge becomes certain about the environment, which can be recognized as very small or no changes in the value function.

2.4 Regret Lower bound for the stochastic Multi-Armed Bandit problems

Lai and Robbins [47] provided asymptotic lower bounds of the expected regret for the stochastic Multi-Armed Bandit problem. In their work, it shows that $R(T) = o(T^a)$ can applies to any strategy for MAB, for all $a > 0$ as $T \to \infty$.

Kaufmann et al.[44] call this condition strongly consistent, since that $\lim_{T\to\infty}\mathbb{E}[S(T)]/T = \mu_{\max}$. When the reward distribution are Bernoulli, for arms i, j, their reward average $\mu_i, \mu_j \in [0,1]$ To define the Kullback-Leibler divergence between two Bernoulli distributions with parameters μ_i and μ_j

$$D_{KL}(\mu_i, \mu_j) = \mu_i \ln \frac{\mu_i}{\mu_j} + (1-\mu_i)\ln \frac{1-\mu_i}{1-\mu_j}$$

The Theorem of asymptotic lower bounds states that

Theorem 2.2. *Distribution-dependent lower bound Consider a strategy that satisfies $R(T) = o(T^a)$ for any set of Bernoulli reward distributions, any arm k with $\Delta_k = \mu^* - \mu_k > 0$, and any $a > 0$. Then, for any set of Bernoulli reward distributions the following holds*

$$\underset{T\to\infty}{lim\ inf} \frac{R(T)}{\ln T} \geqslant \sum_{k\in\mathscr{K}:\Delta_k>0} \frac{\Delta_k}{D_{KL}(\mu_k, \mu^*)} \tag{2.9}$$

Let $N_{k,T}$ be the number of times the strategy pulled arm k up to time T, the bound can be written as,

$$\lim_{T\to\infty} \inf \frac{\sum_{k\in K} \mathbb{E}[N_{k,T}(\mu_{max} - \mu_k)]}{\ln T}$$

They call any strategy that meets this lower bound with equality asymptotically efficient. It's also useful to note that for a strategy that satisfies

$$\lim_{T\to\infty} \frac{\mathbb{E}[N_{k,T}]}{\ln T} \geqslant \frac{1}{D_{KL}(K, \mu_{max})},$$

with equality for all $k \in K$ then refer to the Equation 2.9 satisfy with equality and the strategy is asymptotically optimal.

2.5 Pure Exploration and Best Armed Identification in Multi-Armed Bandit

Bubeck [18, 19] and Chen [24] proposed the same problem that the gambler may sample arms under a given number of times T which may be not necessary to know in advance, and be asked to output an arm recommended. They advocate to evaluate the performance of gambler by the simple regret (refer the section 2.1.1). The distinguish from the classical MAB's modeling is to separate the exploration phase and the exploitation phase. This is the pure exploration problem. Its process is shown as the following modeling,

- Definir the number of rounds T and number of arms K to gambler;
- Parameters the reward distribution $\nu_1, \ldots, \nu_K$ of the arms which is unknown to ganbler;
- Repeat that gambler chooses an arm $k_t \in \mathscr{K}$ and receives the reward $r_{(k_t,t)}$ drawn from ν_{k_t} and independently from the past;
- Finally, the gambler outputs the recommendation k^* arm from the arm set $\mathscr{K}$.

The pure exploration problem is about the design of strategies who makes the best use of the limited budget in order to optimize the performance and identifier their best choice of a decision-making task. Audibert [6] proposed two algorithms to address this problem: UCB-E and Successive Rejects. The former is a highly exploring strategy based on Upper Confidence Bounds, the latter is a parameter-free method based on propressively rejecting the arms which seem to be suboptimal. Audibert shows that these two algorithms are essentially optimal since the regret decrease exponentially at a rate which up to a logarithmic factor.

UCB-E algorithm

This algorithm is the highly exploring policy based on Upper Confidence bounds(UCB-E). When the exploration parameter a (shown in Appendix A.3) is taken of order $\log T$, the algorithm essentially corresponds to the UCB1 algorithm introduced in [11]. And its cumulative regret is of order $\log T$. [18] have shown that algorithms having at most logarithmic cumulative regret, have at least a (non-cumulative) regret of order $T^{-\gamma}$ for some $\gamma > 0$. So taking the parameter a of order $\log T$ is not befitting to reach exponentially small probability of error. It should take a as linear in T can explore much more than ever. The theorem below cited from [6]

Theorem 2.3. *If UCB-E is run with parameter* $0 < a \leqslant \frac{25}{36}\frac{T-K}{H_1}$*, then it satisfies*

$$e_T \leqslant 2TK\exp-\frac{2a}{25}. \tag{2.10}$$

In particular for $a = \frac{25}{36}\frac{T-K}{H_1}$*, we have* $e_T \leqslant 2TK\exp-\frac{T-K}{18H_1}$

Where e_T denote the probabiity of error that the recommendation k_T equals to the optimal arm k^* and H_1 denotes a quantity $H_1 = \sum_{k=1}^{K}\frac{1}{\Delta_k^2}$ (the definition of Δ_k in Section 2.1.1). Theorem 2.3 shows that the probability of error of UCB-E is at most of order $\exp -a$ for $a \geqslant \log T$.

In view of the parameter a if $a \leqslant \frac{25}{36}\frac{T-K}{H_1}$, it can be essentially said: the more it explores, the smaller the regret is. Besides, the smallest upper bound on the probability of error is obtained for a of order T/H_1, and is therefore exponentially decreasing with T. The constant H_1 depends not only on how close the mean rewards of the two best arms are, but also on the number of arms and how close their mean reward is to the optimal mean reward. This constant should be seen as the order of the minimal number n_{k_T} for which the recommended arm is the optimal one with high probability.

Successive Rejects algorithm

This is another algorithm to identifier the best arm in MAB setting by pure exploration. The details are shown in Appendix A.4. Informally it proceeds as follows. First the algorithm divides the time (i.e., the T rounds) in $K-1$ phases. At the end of each phase, the algorithm dismisses the arm with the lowest empirical mean. During the next phase, it pulls equally often each arm which has not been dismissed yet. The recommended arm k_T is the last surviving arm. The length of the phases are carefully chosen to obtain an optimal (up to a logarithmic factor) convergence rate. More precisely, one arm is pulled $n_1 = \left\lceil \frac{1}{\log K}\frac{T-K}{K} \right\rceil$ times, one $n_2 = \left\lceil \frac{1}{\log K}\frac{T-K}{K-1} \right\rceil$ times, …, $n_{K-1} = \left\lceil \frac{1}{\log K}\frac{n-K}{2} \right\rceil$ times. SR does not exceed the budget of T pulls, since, from the definition $\overline{\log}(K) = \frac{1}{2} + \sum_{k=2}^{K}\frac{1}{k}$, we have

$$n_1 + \ldots, + n_{K-1} \leqslant K + \frac{T-K}{\overline{\log}(K)}\left(\frac{1}{2} + \sum_{k=1}^{K-1}\frac{1}{K+1-k}\right) = T$$

For $K = 2$, up to rounding effects, SR is just the uniform allocation strategy.

Theorem 2.4. *The probability of error of SR satisfies*

$$e_n \leqslant \frac{K(K-1)}{2}\exp\left(-\frac{T-K}{\overline{\log}(K)H_2}\right).$$

Here H_2 denotes a quantity $H_2 = \max_{k\in\mathscr{K}} k\Delta_k^{-2}$. The following theorem provides a deeper understanding of how SR works. It lower bounds the sampling times of the arms and shows that at the end of phase k, we have a high-confidence estimation of $\Delta_{(K+1-k)}$ up to numerical constant factor.

Theorem 2.5. *With probability at least* $1-\frac{K^3}{2}\exp\left(-\frac{T-K}{4\overline{\log}(K)}H_2\right)$*, for any arm* k*, we have*

$$n_k(T) \geqslant \frac{T-K}{4\overline{\log}(K)H_2\Delta_k^2} \tag{2.11}$$

With probability at least $1-K^3\exp\left(-\frac{T-K}{32\log(K)H_2}\right)$*, for any* $k \in \{1,\dots,K-1\}$*, the dismissed arm* $l_k = \mathscr{K}_k \setminus \mathscr{K}_{k+1}$ *at the end of phase* k *satisfies*

$$\frac{1}{4}\Delta_{K+1-k} \leqslant \frac{1}{2}\Delta_{l_k} \leqslant \max_{m\in\mathscr{K}_k} \hat{X}_{m,n_k} - \hat{X}_{l_k,n_k} \leqslant \frac{3}{2}\Delta_{l_k} \leqslant 3\Delta_{(K+1-k)} \tag{2.12}$$

The proofs of Theorem 2.4 and 2.5 see in [6].

2.6 Conclusion

This section is the conclusion of all this chapter.

CHAPTER 3

BANDIT WITH SIDE INFORMATION

Contextual Bandit, where the Bandit problem with side information has been introduced in Section 2.1.3. Since it is closely related to work on supervised learning and reinforcement learning, it is usually applied to solve the problem of supervised learning with partial feedback. The classification with partial feedback is a novel and influential problem. It can be traced back to online classification with full feedback and multi-armed bandit learning. This Classification can be considered as a Multi-Armed Bandit problem with side information. Langford [48] extended the Multi-Armed setting to the case where some side information is provided. But, the setting has a high level of abstraction and its application to the classification bandit learning is not straightforward.

In this chapter, we restate the setting in Supervised learning with partial feedback under the frame of Contextual Bandit (in this chapter, it will be called Bandit Feedback), and recall some outstanding researches and contributions. At first, we formally re-introduce some notations in the modeling of Bandit Feedback. Just as what we have previously presented, the problem of Bandit Feedback is composed by two parts: supervised classification learning and partial feedback. For the side of the classification learning, we will focuse on introducing some supervised classification algorithms in the-state-of-art. From other side, we present some important algorithms working with Bandit Feedback, most of them are based on the supervised learning algorithm combining some bandit strategies. After completing the description of the classification with Bandit Feedback, we repose a novel problem, the Multi-Labels classification working with partial feedback. Then, we provide some analysis and some effective algorithm to this issue. Finally, to evaluate the quality of those algorithms, we take some experimentation on comparing them with classical datasets and analyze their respective features and benefits.

3.1 Bandit feedback in Multi-class Classification

Classification is a fundamental task of machine learning, and is by now well understood in its basic variants. Unlike the well-studied supervised learning setting, many recent applications (such as recommender system, ad selection,etc) can not work under the frame of the supervised learning with side information. In this section, we introduce this kind problem: Online Classification with **Bandit Feedback**.

Online classification with bandit feedback, is a bandit variant of the online classification protocol, where the goal is to sequentially learn a mapping from the context space $\mathscr{X} \subseteq \mathbb{R}^d$ to the label space $\mathscr{Y} = \{1,\ldots,K\}$, with $K \geq 2$. In this protocol, forecaster keeps classifiers parameterized $w = (w_1, w_2, \ldots, w_K)$ from the hypothesis space $\mathscr{W} \subseteq \mathbb{R}^{K\times d}$. At each steps $t = 1,2,\ldots,T$, the side information $x_t \in \mathscr{X}$ is sampled as i.i.d., then forecaster predicts the label $\hat{y}_t$, by the linear hypothesis w:

$$\hat{y}_t = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} < w_k, x_t > \tag{3.1}$$

In the standard online protocol, forecaster observes the true label y_t associated with x_t after each prediction and uses this full information to adjust the classifier w_t. However, in the bandit version, forecaster only observes an indicator $\mathbb{1}(\hat{y}_t = y_t)$, that is , whether the prediction at time t is correct or not. We denote the cumulative loss of supervised learning as following:

$$L = \sum_{t=1}^{T} l\,(w_t; (\mathbf{x}_t, y_t, \hat{y}_t)) \tag{3.2}$$

And the one with Bandit Feedback is defined as below:

$$L_{BF} = \sum_{t=1}^{T} l_{BF}\,(w_t; (\mathbf{x}_t, \mathbb{1}(\hat{y}_t = y_t)) \tag{3.3}$$

3.1.1 Multiclass Classification

Multiclass Classification is a problem of classifying the samples into several different classes and online learning is performed as a sequence of trials experiment. To solve this problem, the algorithms are aiming at learning a predictor $h : \mathscr{X} \to \mathscr{Y}$, which maps the instances to the classes space. The simplest approach to tackle multiclass prediction problem is by reduction from multiclass classification to binary classification. That is the methods we often mention: One-versus-All and All-versus-All. See Figure 3.1, there are some points classified into three classes (classified as their colors), by the method One vs All, it's necessary to find three binary classifiers who can only identify one class see Figure 3.2. Crammer has introduced several additive and multiplicative algorithms in [28], where Perceptron [58] and Winnow [49] are two such important algorithms. Much analysis has been done, Kivinen and Warmuth developed potential functions that can be used to analyze different online algorithm [45].

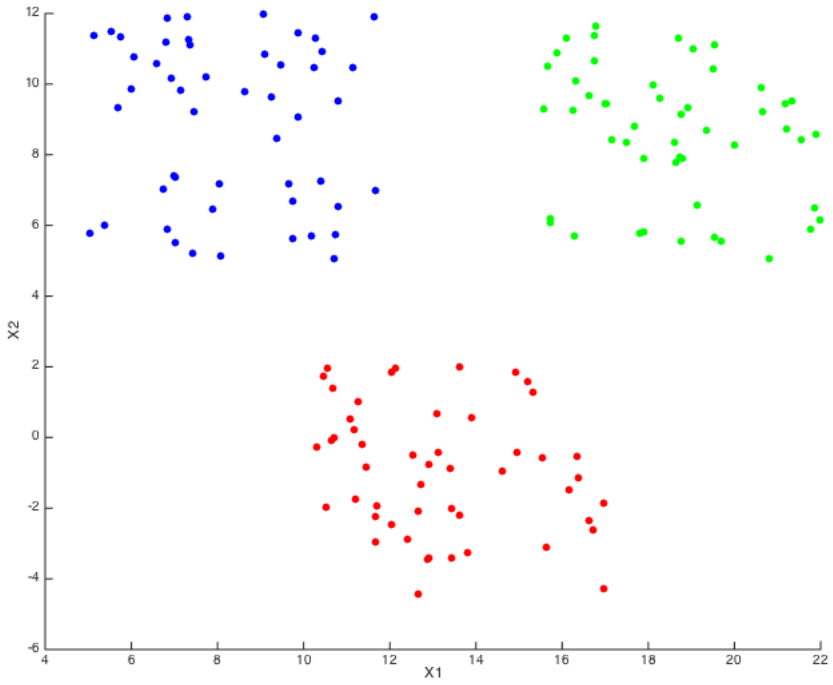

Figure 3.1: Multiclass task

For the process of multiclass classification online learning, at trial t, the algorithm firstly samples an instance $x_t \in \mathscr{X} \subseteq \mathbb{R}^d$ and classifies it into the set of all possible labels constitutes a finite set denoted by $\mathscr{Y}$. Mostly, the algorithms of online classification assume that $\mathscr{Y}$ is known in advance. We denote the set of unique labels observed on rounds from 1 to $t-1$ by Y_t (In general, this set is known in advance and non-changeable during all rounds, so it could be simplified by Y). After predicted the label $\hat{y}_t$ by the algorithm of online learning, the true class $y_t \in Y_t$ is revealed. Sometimes, the prediction $\hat{y}_t$ is different to the true label y_t, in that case, it makes an incorrect prediction. To evaluate the accuracy or quality of classifiers, the prediction mistakes (denoted as M), regret (or instantaneous loss, as l) or cumulative regret (denoted as R or L) come into being. To be computed on each round, specially when $\hat{y}_t \neq y_t$.

The goal of all classification algorithms, is to minimize the total number of prediction mistakes or cumulative regret. To achieve this goal, the algorithms should update the classifiers by supervised learning mechanism at the end of each trial.

Here, we make some definitions: the feature vector representation $\Phi(\mathbf{x}, y)$ induced by the instance label pair $(\mathbf{x}, y)$. Here, $\Phi(\mathbf{x}, y)$ is a $K \times d$ matrix which is composed of K blocks of feature size d. All blocks but the y'th block of $\Phi(\mathbf{x}, y)$ are set to be the zero vector while the y'th blocks is set to be $\mathbf{x}$. Applying a single prototype multi-class algorithm to this problem produces a hypothesis $w \in \mathbb{R}^{K \times d}$ from $\mathscr{W}$ on every online round. Analogous to the construction

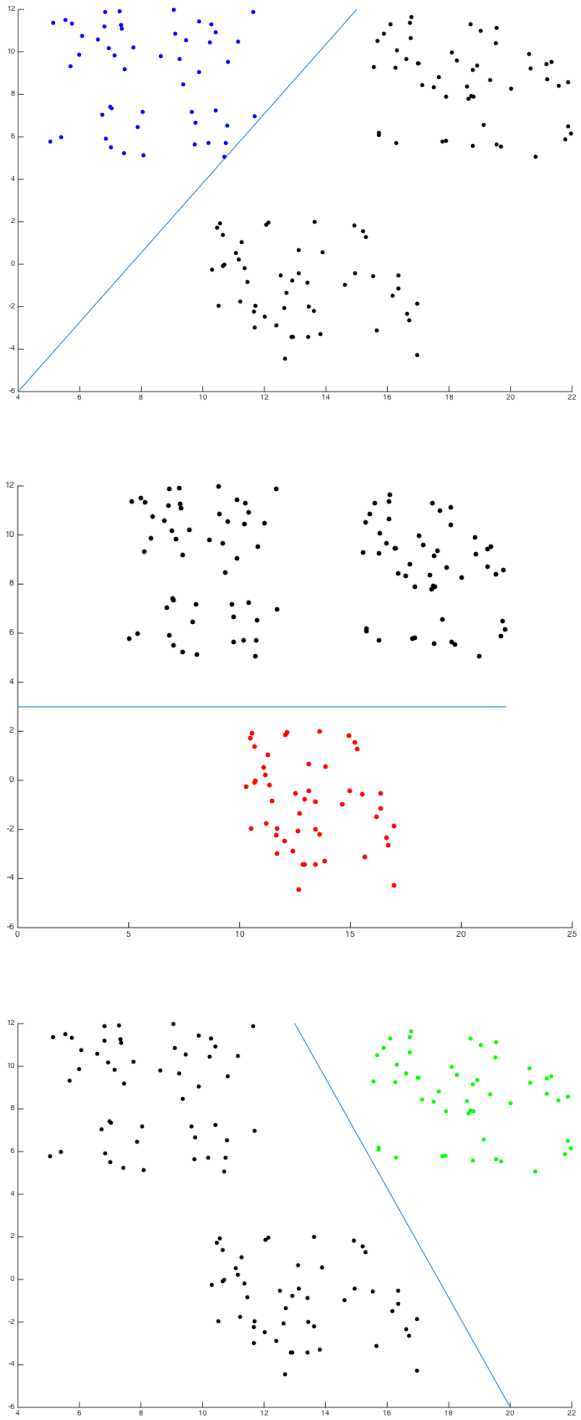

Figure 3.2: Reduction from multiclass classification to binary classification (One vs All)

of $\Phi(\mathbf{x}, y)$, w is composed of K blocks of size d and denote block r by w_r. By construction, we get that $< w \cdot \Phi(x_t, r) >=< w_r, x_t >$. Recall the hinge loss in binary classification, it's defined to be $[1 - y_t < w, \mathbf{x}_t >]_+$, where $[a]_+$ means $\max(a, 0)$. The hinge loss in the multiclass frame as follows:

$$l(w_t;(\mathbf{x}_t, y_t, \hat{y}_t)) = [1 + < w_t, \Phi(\mathbf{x}_t, \hat{y}_t) > - < w_t, \Phi(\mathbf{x}_t, y_t) >]_+ \tag{3.4}$$

After introduce the principal multiclass definitions, in the following of this section, we study some approaches for learning multiclass classifiers. Most of them are linear multiclass predictors.

Perceptron was proposed by F. Rosenblatt [58]. It is a general computational model with some numerical weights. By Minsky and Papert [55], it was refined and perfected in 1960s. For the simplest binary model, the input space $\mathscr{X} \subseteq \mathbb{R}^d$ can be separated into two sets P and N. The algorithm Perceptron is to look for a weight vector $w \in \mathbb{R}$ with a product function h, where $h(x_t) =< w, x_t > \in \mathbb{R}$. If the P and N are linear separated, the ideal weighted vector be assumed that all points of P holds the product $h(x_P) > 0$ and all points of N holds $h(x_N) < 0$.

Generally, it initiates to choose the weight vector w_0 randomly. For the binary state, if the instance on round t^{th} x_t belongs to the set P, y_t the class of instance x_t is $+1$, else y_t equals to -1. Then, $y_t \cdot h(x_t) > 0$, it means the weight vector make a good prediction, and continue to predict a new instance; from the opposite direction, when it predicts wrong, the value of product $y_t \cdot h(x_t) < 0$, it should take an update to the weight vector with the instance vector x_t

$$w_{t+1} = \begin{cases} w_t & \text{if } y_t \cdot h(x_t) > 0 \\ w_t + y_t \cdot x_t & \text{else} \end{cases}$$

The Perceptron in multiclass classification with K classes, it should look for a set of K weight vectors $W = (w_1, w_2, \ldots, w_K) \in \mathscr{W} \subseteq \mathbb{R}^{K \times d}$. The prediction way should be referred to the Function 3.1. However, Perceptron updates according to the incorrect classification for each instance. It does not benefit the global information by its instantaneous strategy, so it is a greedy, local algorithm, and to be lead to an exponential number of updates with data non-separable. Its mistake bound convergence in $(R/\gamma)^2$ where $R = \max \parallel x_t \parallel$ and $\gamma = \min| < u, x_t > |$ with ideal classifiers u. More details of Perceptron see Appendix A.5.

Second-order Perceptron[21]. In the previous part, we talked about a popular, local and greedy linear algorithm– Perceptron. Its serious problem is with incremental correlation. Here, we address a second order variant of Perceptron, which is proposed by Cesa-Bianchi. In this algorithm, there is a correlation matrix $S_t = \sum_{s=1}^{t} x_s x_s^T \in \mathbb{R}^{d \times d}$. With S_t^{-1}, the correlation matrix is reduced to the identity matrix I_t.

In the basic form, Second-order Perceptron algorithm takes an input parameter $a > 0$. to compute its prediction in trial t the algorithm uses an d-row matrix S_{t-1} and an K-dimensional weight vector w_{t-1}, where subscript $t-1$ indicates the number of times matrix S and vector w have been updated in the first $t-1$ trials. Initially, the algorithm sets $S_0 = \mathbf{0}$ and

$w_0 = \mathbf{0}$. Upon receiving the t^{th} instance x_t, the algorithm predicts the label of x_t with $\hat{y}_t = \arg\max_{i\in\{1,\ldots,K\}} \left((\sum_{s=1}^{t-1} y_t x_s)^T (aI_d + S_t S_t^T)^{-1} x_t\right)$, with I_d being the $d \times d$ identity matrix. If $\hat{y}_t \neq y_t$, then a mistake occurs and the algorithm updates, if $\hat{y}_t = y_t$, no update takes place, and hence the algorithm is mistake driven.

The second-order Perceptron algorithm retains the properties of sparsity and efficient dual variable representation. This allows us to efficiently run the algorithm in any reproducing Kernel Hillbert space for the non-linear situation. By introducing the second-order matrix, Second Order Perceptron can effectively reduce the misclassification around the boundary and assumed to get a mistake upper bound. Confidence weighted can maintain each feature with a different confidence level, in order that the features with low degree of confidence need to update than the one with high level. To minimize KL divergence, the confidence weighted update method to ensure that the probability of each new sample can be correctly classified never be less than a fixed parameter.

Passive-Aggressive Algorithm[26] is an effective framework for performing max-margin online learning. Here, we address Online Passive-Aggressive algorithms, who learns the predictors from the linear hypothesis space during an online sequential.

Here, all definitions are consistent with other sections. $\gamma(w_t;(x_t,y_t))$ is used to present the margin between each labels.

$$\gamma(w_t;(x_t,y_t)) = < w_t, \Phi(x_t,y_t) > - \max_{s\neq y_t} < w_t \cdot \Phi(x_t,s) >$$

.

The margin is positive only if the relevant label is with higher density than all of the other irrelevant labels. With the definition of margin, it computes an instantaneous loss by hinge-loss function as following,

$$l(w;(x,Y)) = \begin{cases} 0 & \gamma(w;(x,Y)) \geqslant 1 \\ 1 - \gamma(w;(x,Y)) & \text{otherwise} \end{cases} \tag{3.5}$$

The PA update rule is derived by defining the new weight w_{t+1} as the solution to the optimization problem:

$$w_{t+1} = \underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \parallel w - w_t \parallel^2 \quad s.t. \;\; (w;(x_t,Y_t)) = 0. \tag{3.6}$$

Intuitively, if w_t suffers no loss from the new instance, that the hinge loss $l_t(w_t;(\mathbf{x}_t,y_t))$ equals to 0, the algorithm passively assigns $w_{t+1} = w_t$; otherwise, it aggressively makes the new classifiers w_{t+1} satisfy that $l_t(w_{t+1};(\mathbf{x}_t,y_t))$ attain to no loss.

The single constraint that choose to satisfy is $w \cdot \Phi(x_t,r_t) - w \cdot \Phi(x_t,s_t) \geqslant 1$ and thus w_{t+1} is set to be the solution of the following simplified constrained optimization problem,

$$w_{t+1} = \underset{w}{\operatorname{argmin}} \frac{1}{2} \parallel w - w_t \parallel^2 \quad s.t. \;\; w \cdot (\Phi(x_t,r_t) - \Phi(x_t,s_t)) \geqslant 1 \tag{3.7}$$

.

The apparent benefit of this simplification lies in the fact that Eq. 3.7 has a closed form solution. To draw the connection between the multilabel setting and binary classification, considering of the vector $\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)$ as a virtual instance of a binary classification. Therefore, the closed form solution of Eq. 3.7 is

$$w_{t+1} = w_t + \tau_t(\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)) \tag{3.8}$$

, with,

$$\tau_t = \frac{l_t}{\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \|^2}$$

Although it's essentially neglecting all but two labels on each step of the multiclass update, it can still obtain multiclass cumulative loss bounds. The key observation in the analysis it that,

$$l(w_t;(x_t, y_t)) = [< w_t, \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) > +1]_+$$

.

For the bounds of multiclass PA algorithm, it needs to cast the assumption that for all t it holds that $\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \| \leqslant R$. This bound can immediately be converted into a bound on the norm of the feature set since $\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \| \leqslant \| \Phi(x_t, y_t) + \| \Phi(x_t, \hat{y}_t) \|$. Thus, if the norm of the mapping $\Phi(x_t, k)$ is bounded for all t and $k \in \mathscr{Y}$ then so is $\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \|$. And the bounds on the cumulative loss of the algorithms is relative to the smallest loss that can be attained by any fixed hypothesis.

3.1.2 The algorithms of multiclass classification with Bandit Feedback

In the conventional supervised learning paradigm, the forecaster has access to a data set in which the true labels of the inputs are provided. Sometime, the environment just provide the partial feedback instead of full one. Such problems are natural being, it's the bandit versions of multiclass prediction problems.

Here, we denote the number of classes is K, and by $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ the sequence of training examples received over trials, where $x_i \in \mathbb{R}^d$ and T is the number of training instances. In each trial, we denote the prediction by $\hat{y}_t \in \{1, \ldots, K\}$. Unlike the classical setup of online learning where an oracle provides the true class label $y_i \in \{1, \ldots, K\}$ to the learner, in the case of partial feedback, the learner only receives one bit to response whether the prediction equals to the true label, i.e., $\mathbb{1}[y_t = \hat{y}_t]$. The Bandit feedback is an application of Contextual Bandit with side information. So the goal of this problem is not only to minimize the error bound, but also to keep balance between Exploration and Exploitation. Some popular strategies of this issue have been introduced in Section 2.3, i.e. UCB, Thompson, ϵ-greedy etc. To apply trade-off strategies to this issue, we should understand the relationship between the prediction $\hat{y}_t$ and the label set $\mathscr{Y}$. The prediction $\hat{y}_t$ is the result of exploitation by the past information, it is the optimal arm or

sub-optimal depends on the hypothesis. Unlike the supervised learning, we have no knowledge about the true label of y_t. So, it's necessary to sample other labels to explore more information of the label setting.

In this section, we will introduce a few traditional multiclass classification on combining the Bandit policies.

Banditron [43] (see in Appendix A.8), is a simple but effective learning strategy for online classification with bandit feedback, which is based on the algorithm Perceptron. Despite its age and simplicity, the Perceptron has proven to be quite effective in practical problems (more details about Perceptron see the previous section or [58]).

Similar to the Perceptron, at each round, the prediction $\hat{y}_t$ can be the best label according to the current weight matrix w, i.e. $\hat{y}_t = \underset{i\in\{1,\dots,K\}}{\operatorname{argmax}} < w_{t,i}, x_t >$. Mostly, Banditron exploits the quality of the current weight matrix to predict the label $\hat{y}_t$. Unlike the Perceptron, if $\hat{y}_t \neq y_t$, then it's difficult to make an update since it's blind to the identity of y_t. Roughly speaking, it is difficult to learn when to exploit using w_t . Since that, on some of rounds it's necessary to let the algorithm explore and uniformly predict a random label from the label set $\mathscr{Y}$. It's denoted by $\tilde{y}_t$ the predicted label. On rounds, in which it explores, (where $\tilde{y}_t \neq \hat{y}_t$), if the forecaster additionally receives a positive feedback, i.e. $\tilde{y}_t = y_t$, then it indirectly obtains the full information regarding the identity of y_t, therefore it could update the weight matrix using this positive instance. The parameter ϵ controls the exploration-exploitation tradeoff, this is the ϵ-Greedy strategy (refer to the Section 2.3.4.

The above intuitive argument is formalized by defining the update matrix $\tilde{U}_t$ to be a function of the randomized prediction $\tilde{y}_t$. We emphasize that $\tilde{U}_t$ accesses the correct label y_t only through the indicator $\mathbb{1}[y_t = \tilde{y}_t]$ and is thus adequate for the bandit setting. Kakade[43] show that the expected value of the Banditron's update matrix $\tilde{U}_t$ is exactly the Perceptron's update matrix U_t. Banditron, the linearpredictor with ϵ-Greedy strategy in bandit setting could be bounded for the regret bound $O(T^{2/3})$ compared to the hinge loss.

Confidit[27], with the different strategy of tradeoff to Banditron, Confidit uses an alternative approach UCB(see in Section 2.3.3), which is to maintain additional confidence information about the predictions. Specifically, given an input $\mathbf{x}_t$, the algorithm not only computes score values, but also non-negative uncertainty values for these scores, denotes by $\epsilon_{i,t}$ an upper bound of confidence interval. Intuitively, high values of $\epsilon_{i,t}$ indicate that the algorithm is less confident in the value of the score $w_i^T \mathbf{x}_t$. Given a new example, the algorithm outputs the label with the highest upper confidence bound (UCB), computed as the sum of score and uncertainty as following,

$$\hat{y}_t = \underset{i\in\{1,\dots,K\}}{\operatorname{argmax}}(\mathbf{w}_i^T \mathbf{x}_t + \epsilon_{i,t}).$$

Intuitively, a label $\hat{y}_t$ is output by the algorithm if either its score is high or the uncertainty in predicting is high, and there is necessary to obtain information about it. Specifically, this

algorithm is based on the Second Order Perceptron, it maintains is a positive semi-definite matrix per label, $A_{i,t} \in \mathbb{R}^{d \times d}$ to compute the upper confidence to each label. More details of this algorithm described in Appendix A.9. Confidit develops the Second Order Perceptron with UCB strategy to solve the supervised problem with Bandit feedback. And it uses the correlation matrix to estimate the uncertainty of label set. Its regret bound of $O(\sqrt{T}\log T)$, which is more excellent than the one of Banditron.

3.2 Bandit feedback in multi-labels

After introduce the bandit feedback in multiclass, here we address another kind of bandit feedback: the Bandit Feedback in Multi-labels Classification. This problem exit widely in the recommendation system, i.e. a limited number of banners placed at different positions on a web-page. The system's goal is to select several advertisements that the user maybe feel interest in. Just like the problem of multiclass with bandit feedback, this one can neither observe the user's other favorites. So it is collectively referred to as learning with bandit feedback. As opposed possible response, in the partial feedback setting, the system only observes the response to very limited options and, specifically, the option that was actually recommended.

We consider instantiations of this problem in the multilabel setting. Learning proceeds in rounds, in each time step t the algorithm receives an instance $\mathbf{x}_t$ and outputs an ordered subset $\hat{Y}_t$ of labels from a finite set of possible labels $\mathscr{Y} = \{1, 2, \ldots, K\}$. Restrictions might apply to the size of $\hat{Y}_t$. This set corresponds to the aforementioned recommendations, and is intended to approximate the true set of preferences associated with $\mathbf{x}_t$. However, the latter set is never observed. In its stead, the algorithm receives $Y_t \cap \hat{Y}_t$, where $Y_t \subseteq \mathscr{Y}$ is a noisy version of the true set of user preferences on $\mathbf{x}_t$. When we are restricted to $|\hat{Y}_t| = 1$ for all t, the problem becomes a familiar problem: the multiclass classification with bandit feedback.

3.2.1 Multilabel Classification

This section, we introduce a form of supervised learning: Multi-label Learning. In supervised learning, multiclass classification is a common learning problem where the goal is to learn from a set of instances, each associated with a unique class label from a set of disjoint class labels $\mathbf{L}$. Depending on the total number of disjoint classes in $\mathbf{L}$, the problem can be identified as binary classification ($|\mathbf{L}| = 2$) or multiclass classification (when $|\mathbf{L}| > 2$) . Unlike these problems, multi-label classification is to learn from a set of instances where each instance belong to one or more classes in $\mathbf{L}$.

Here, let's define some notations of multi-label classification. Let $\mathscr{X}$ be an instance space, and $\mathbf{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ be a finite set of class labels. An instance $\mathbf{x}_t \in \mathscr{X}$, represented in terms of features vector $\mathbf{x}_t = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, is associated with a subset of labels $\mathbf{L} \in 2^{\mathbf{L}}$. Notice that we call

this set $\mathbf{L}$ be the set of relevant labels of $\mathbf{x}_t$. Denote this relevant labels set $\mathbf{L}$ with a binary vector $Y = (y_1, y_2, \ldots, y_K)$, where $y_i = 1 \Leftrightarrow \lambda_i \in \mathbf{L}$. $\mathscr{Y} = \{0,1\}^K$ is the set of all such possible labelings.

Given a training set, $S = (x_i, Y_i)$, $1 \leqslant i \leqslant T$, consisting T training instances, $(x_i \in \mathscr{X}, Y_i \in \mathscr{Y})$ i.i.d. drawn from an unknown distribution D, the goal of the multi-label learning is to produce a multi-label classifier $h : \mathscr{X} \to \mathscr{Y}$ that optimizes some specific evaluation function (i.e. loss function).

Refer to [60], it presents a number of very simple transformation methods which actually transform multi-label data in such a way so that existing classification algorithm (i.e. binary classifiers) can be applied. It would be easier to describe such algorithms using an example multi-label data in Table 3.1. Notice that the instances features are ignared here, because they are not really important for describing these algorithms. There are four instances belong to at least one of the 4 classes, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

Table 3.1: Example of multi-label dataset

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

The copy transformation method replaces each example $(\mathbf{x}_t, Y_t)$ with $|Y_t|$ examples $(\mathbf{x}_t, \lambda_t)$, for each $\lambda_i \in Y_t$. An extension to this is to use an weight of $\frac{1}{|Y_t|}$ to each of these newly created examples. This is called dubbed copy-weight method.

For each instance, the select family of transformation methods replaces Y_t by one of its members. Depending on how this one member is selected, there can be several versions, namely, select-min (least frequent), select-max (most frequent), and select-random (randomly selected).

Finally, ignore transformation simply ignores the multilabel instances and runs the training with single label instances only.

Label Powerset (LP) Label Powerset(LP) is a straight forward method that considers each unique set of labels in a multilabel training data as one class in the new transformed data. Therefore, the new transformed problem is a single label classification task. For a new instance, LP outputs the most probable class which actually is a set of classes in the original data. It is also possible to produce a ranking of labels using LP, given the classifier can output a probability distribution over all newly formed classes. Table 3.2 shows the dataset transformed using LP method. Notice that the computational complexity of LP is upper bounded by $min(n, 2^K)$, where n is the total number of data instances and K is the total number of classes in the training data. In practice complexity would be much less than 2^K, still for large values of T and K this can be an issue. The second problem with this approach is that, a large number of classes would be associated with very few examples and that would also pose extreme class imbalance problem for

learning. **Binary Relevance (BR)** Binary Relevance (BR) is one of the most popular approaches

Table 3.2: Transformed data using Label Power-set method

Instance	Label
1	$\lambda_{2,3}$
2	λ_1
3	$\lambda_{1,2,3}$
4	$\lambda_{2,4}$

as a transformation method that actually creates K datasets ($K = |\mathbf{L}|$, total number of classes), each for one class label and trains a classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset D_{λ_j}, $1 \leqslant j \leqslant K$ positively labels instances that belong to class λ_j and negative otherwise. Table 3.3 show the example dataset transformed for BR.

Table 3.3: Transformed data produced by Binary Relevance (BR) method

Instance	Label
1	$\neg\lambda_1$
2	λ_1
3	λ_1
4	$\neg\lambda_1$

((a))

Instance	Label
1	λ_2
2	$\neg\lambda_2$
3	λ_2
4	λ_2

((b))

Instance	Label
1	λ_3
2	$\neg\lambda_3$
3	λ_3
4	$\neg\lambda_3$

((c))

Instance	Label
1	$\neg\lambda_4$
2	$\neg\lambda_4$
3	$\neg\lambda_4$
4	λ_4

((d))

Once these datasets are ready, it is easy to train one binary classifier for each. For any new instance, BR outputs the union of the labels λ_j that are positively predicted by the K classifiers. While BR has been used in many practical applications, it has been widely criticized for its implicit assumption of label independence which might not hold in the data.

Ranking by Pairwise Comparison (RPC) [16] transforms the multilabel dataset into $\binom{k}{2}$ binary label datasets, one for each pair of labels, (λ_i, λ_j), $1 \leqslant i < j \leqslant K$. Each dataset retains the instances from the original dataset that belong to at least one of the corresponding labels but not both (show in Table 3.4).

Table 3.4: Transformed data by RPC method

$\mathbf{x}_t$	L
1	$\lambda_{\neg 1,2}$
2	$\lambda_{1,\not 2}$
4	$\lambda_{\neg 1,2}$

((a))

$\mathbf{x}_t$	L
1	$\lambda_{\neg 1,3}$
2	$\lambda_{1,\not 3}$
4	$\lambda_{\neg 1,\neg 3}$

((b))

$\mathbf{x}_t$	L
1	$\lambda_{1,\neg 4}$
2	$\lambda_{1,\not 4}$
4	$\lambda_{\neg 1,4}$

((c))

$\mathbf{x}_t$	L
4	$\lambda_{2,\neg 3}$

((d))

$\mathbf{x}_t$	L
1	$\lambda_{2,\neg 4}$
3	$\lambda_{2,\not 4}$

((e))

$\mathbf{x}_t$	L
3	$\lambda_{3,\neg 4}$
4	$\lambda_{\neg 3,4}$

((f))

A binary classifier is then trained on each of these datasets. Also, given a new instances, it is

easy to obtain a ranking of labels by first invoking all these binary classifiers and then counting their votes for each label. Mencia[50] proposes Multi-label Pairwise Perceptron (MLPP) algorithm that uses RPC with perceptrons as the binary classification method.

Calibrated Label Ranking (CLR) Even though ranking provides a relative order of the labels, Furnkranz[50] argues that such ranking does not have a natural "zero-point" and therefore, does not provide any information about the absolute preference that can distinguish among all alternatives. It could be misleading to distinguish between the sets of relevant and non-relevant classes based on the label ranking. CLR was proposed in that situation by Furnkranz that is an extension of RPC, introducing an additional label to the original label set, which can be interpreted as a "neutral breaking point" and can be thought as a split point between relevant and irrelevant labels. Thus a calibrated ranking,

$$\lambda_{i1} \succ \lambda_{i2} \succ \cdots \succ \lambda_{ij} \succ \lambda_0 \succ \lambda_{i(j+1)} \succ \cdots \succ \lambda_{iK}$$

clearly is a ranking of the labels (ignore the calibration label λ_0) and at the same time creates a bipartition of relevant ($\lambda_{i1} \ldots \lambda_{ij}$) and irrelevant ($\lambda_{i(j+1)} \ldots \lambda_{iK}$) labels. Each example that is annotated with a particular label, clearly is a positive example for that label and is treated as a negative example for the calibration label. Each example that is not annotated with a label is clearly a negative example for that label and is treated as a positive example for the calibration label. Thus a Binary Relevance (BR) classifier can then be employed to discriminate between the calibrated label and each of the other labels. Intuitively, while applied to the dataset in Table 3.1, CLR could work on both the data in Table 3.3 and 3.4, the latter one is for the calibration label.

3.2.2 Some multi-label algorithms working in Bandit setting

There are so many literature to introduce the multilabel classification [35, 46, 52, 60]. Such problems are collectively referred to as learning with full information. As opposed to the full information case, in this section, we consider the multilabel classification follows the bandit setting. Firstly, it should construct a model for this setting. At t time the side information vector $\mathbf{x}_t \in \mathbb{R}^d$, is allowed to output at a subset $\hat{Y}_t \subseteq [K]$ of the set of possible labels, then the subset of labels $Y_t \subseteq [K]$ associated with $\mathbf{x}_t$ is generated, and get a bandit feedback $\hat{Y}_t \cap Y_t$.

The algorithm based on 2nd-order descent in Bandit feedback Here address the algorithm based on 2nd-order descent method, which is proposed by Gentile[25]. It uses a linear predictor with a cost-sensitive multilabel loss that generalized the standard Hamming loss. In such setting, it is proven that the cumulative regret is bounded by $T^{1/2} \log T$. The loss suffered by the algorithm may take into account several things: the distance between Y_t and $\hat{Y}_t$, as well as the cost for playing $\hat{Y}_t$. The cost $c(\hat{Y}_t)$ associated with $\hat{Y}_t$ might be given by the sum of costs suffered on each class $i \in \hat{Y}_t$, where we possibly take into account the order in which i occurs within $\hat{Y}_t$. Specifically, given constant $a \in [0,1]$ and costs $c = \{c(i,s), i = 1, \ldots, s, s \in [K]\}$, such that

$1 \geqslant c(1,s) \geqslant c(2,s) \geqslant \cdots \geqslant c(s,s) \geqslant 0$, for all $s \in [K]$, this algorithm considers the loss function like below:

$$l_{a,c}(Y_t, \hat{Y}_t) = a|Y_t \setminus \hat{Y}_t| + (1-a) \sum_{i \in \hat{Y}_t \setminus Y_t} c(j_i, |\hat{Y}_t|)$$

, where j_i is the position of class i in $\hat{Y}_t$, and $c(i_i, \cdot)$ depends on $\hat{Y}_t$ only through its size $|\hat{Y}_t|$.

Working with the above loss function makes the algorithm's output $\hat{Y}_t$ become a ranked list of classes, where ranking is restricted to the deemed relevant classes only. In our setting, only a relevance feedback among the selected classes is observed (the set $Y_t \cap \hat{Y}_t$), but no supervised ranking information is provided to the algorithm within this set. Alternatively, we can think of a ranking framework where restrictions on the size of $\hat{Y}_t$ are set by an exogenous parameter of the problem, and the algorithm is required to provide a ranking complying with these restrictions.

Let $\mathbb{P}_t(\cdot)$ be a shorthand for the conditional probability $\mathbb{P}_t(\cdot|\mathbf{x}_t)$, where the side information vector $\mathbf{x}_t$ can in principle be generated by an adaptive adversary as a function of the past. Then $\mathbb{P}_t(y_{1,t}, \ldots, y_{K,t}) = \mathbb{P}(y_{1,t}, \ldots, y_{K,t}|\mathbf{x}_t)$, where the marginals $\mathbb{P}_t(y_{i,t} = 1)$satisfy

$$\mathbb{P}_t(y_{i,t} = 1) = \frac{g(-\mathbf{u}_i^T \mathbf{x}_t)}{g(\mathbf{u}_i^T \mathbf{x}_t) + g(-\mathbf{u}_i^T \mathbf{x}_t)}, i = 1, \ldots, K \tag{3.9}$$

for some K vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^d$ and some function $g : D \subseteq \mathbb{R} \to \mathbb{R}^+$. The model is well defined if $\mathbf{u}_i^T \mathbf{x}_t \in D$ for all i and all $\mathbf{x}_t \in \mathbb{R}^d$ chosen by the adversary. We assume for the sake of simplicity that $\| \mathbf{x}_t \| = 1$ for all t. Notice that at this point the variables $y_{i,t}$ need not be conditionally independent. We are only defining a family of allowed joint distributions $\mathbb{P}_t(y_{1,t}, \ldots, y_{K,t})$ through the properties of their marginals $\mathbb{P}_t(y_{i,t})$.

The algorithm in Appendix A.10, here the unknown model vectors $u_1, \ldots, u_K$ with prototype vectors $w'_{1,t}, \ldots, w'_{K,t}$, being $w'_{i,t}$ the time t approximation to u_i, satisfying similar constraints we set for the $_i$ vectors. For the sake of brevity, we let $\hat{\Delta}'_{i,t} = \mathbf{x}_t^T w'_{i,t}$, and $\Delta_{i,t} = u_i^T \mathbf{x}_t, i \in [K]$. The algorithm uses $\hat{\Delta}'_{i,t}$ as proxies for the underlying $\Delta_{i,t}$ according to the upper confidence approximation scheme $\Delta_{i,t} \approx [\hat{\Delta}'_{i,t} + \epsilon_{i,t}]_D$, where $\epsilon_{i,t} \geqslant 0$ is a suitable upper confidence level for class i at time t. The algorithm's prediction $\hat{Y}_t$ at time t has the same form as the computation of the Bayes optimal sequence Y_t^* where we replace the $g(-\Delta_{i,t})$ of $p_{i,t}$ by $g([\hat{\Delta}'_{i,t} + \epsilon_{i,t}]_D)$.The algorithm receives in input the loss parameters a and $c(i,s)$, the model function $g(\cdot)$ and the associated margin domain $D = [-R, +R]$, and maintains both K weight vector $w_{i,t} \in \mathbb{R}^d$. At each time step t, upon receiving the d-dimensional instance vector $\mathbf{x}_t$ the algorithm uses the weight vectors $w_{i,t}$ to compute the prediction vectors. These vectors can easily be seen as the result of projecting $w_{i,t}$ onto the space of w where $|w^T \mathbf{x}_t| \leqslant R$, the distance function $d_{i,t-1}$, i.e. $w'_{i,t} = \arg\min_{w \in \mathbb{R}^d : w^T \mathbf{x}_t \in D} d_{i,t-1}(w, w_{i,t}), i \in [K]$, where $d_{i,t}(u,w) = (u-w)^T A_{i,t}(u-w)$. Vectors $w'_{i,t}$ are then used to produce prediction values $\hat{\Delta}'_{i,t}$ involved in the upper confidence calculation of $\hat{Y}_t \subseteq [K]$. Next, the feedback $Y_t \cap \hat{Y}_t$ is observed, and the algorithm in Appendix A.10 promotes all classes $i \in Y_t \cap \hat{Y}_t$, denotes all classes $i \in \hat{Y}_t \setminus Y_t$(sign $s_{i,t} = -1$), and leaves all remaining classes $i \neq \hat{Y}_t$ unchanged (sign $s_{i,t} = 0$). The update $w'_{i,t} \to w_{i,t+1}$ is based on the gradients $\nabla_{i,t}$ of a

loss function $L(\cdot)$ satisfying $L'(\Delta) = -g(\Delta)$. On the other hand, the update $A_{i,t-1} = A_{i,t}$ uses the rank one matrix $\mathbf{x}_t\mathbf{x}_t^T$. In both the update of $w'_{i,t}$ and the one involving $A_{i,t-1}$, the reader should observe the role played by the signs $s_{i,t}$.

3.3 Conclusion

CHAPTER 4

MULTI-OBJECTIVE MULTI-ARMED BANDIT

Multi-armed bandits is a machine learning paradigm used to study and analyze resource allocation in stochastic and uncertain environments. The multi-armed bandit problem that considers reward vectors and imports techniques from multi-objective optimization into the multi-armed bandits algorithms is referred to as multi-objective multi-armed bandits(MOMAB). In this chapter, we first introduce the MOO's problem,after that, we will talk about the MOMAB.

4.1 Multi-Objective Optimization

The simultaneous optimization of two or more conflicting objectives is called Multi-Objective Optimization(MOO). Practical optimization problems, especially the engineering design optimization problems, often have a multi-objective nature. For example, in the engineering design problem, some structural performance criteria are to be maximized, while the weight of the structure and the implementation costs should be minimized simultaneously.

An MOO problem with n decision variables $x_1,\ldots,x_n$ and d objectives $f_1,\ldots,f_d$ is formulated as follows:

$$\begin{aligned} \text{Optimize} y = \mathbf{f}(\mathbf{x}) &= (f_1(x),\ldots,f_d(x)) \\ s.t. \mathbf{x} &= (x_1,\ldots,x_n) \in \mathscr{X} \end{aligned} \tag{4.1}$$

where the decision vector $\mathbf{x}$ ranges in the decision (parameter) space $\mathscr{X}$, $\mathbf{y}$ is the objective vector with f_i mapping $\mathscr{X}$ onto $\mathbb{R}$, $\mathbb{R}^d$ is the objective space. The objective function is the mapping $\mathbf{f}: \mathscr{X} \to \mathbb{R}^d$. In the following, without loss of generality, we only consider objectives $\mathbf{f_i}$, $\mathbf{i} = \mathbf{1},\mathbf{2},\ldots,\mathbf{d}$ to be maximized.

4.1.1 Front Pareto setting

There are many ways to formulate an MOO problem. The key question in MOO regards the trade-off between the conflicting objectives f_i, $i = 1,2,\ldots,d$. This subsection concentrates on the concept of Pareto optimization at the core of Multi-Objective Optimization, originated from the engineer and economist Vilfredo Pareto[56], stating that:

> Multiple criteria solutions could be partially ordered without making any preference choices a prior.

Several notions related to Pareto optimality are frequently used in the MOO literature as the following, referring the reader to [30] for more detail.

Definition 4.1. Weak Pareto dominance Given two objective vectors $\mathbf{y} = (y_1,\ldots,y_d)$,$\mathbf{y}' = (y'_1,\ldots,y'_d)$, $\mathbf{y}$ is said to weakly dominate $\mathbf{y}'$(noted $\mathbf{y} \succeq \mathbf{y}'$) iff $y_i \geqslant y'_i, \forall i \in [1,\ldots,d]$.

Definition 4.2. Pareto dominance Objective vector $\mathbf{y}$ dominated objective vector $\mathbf{y}'$ (noted $\mathbf{y} \succ \mathbf{y}'$) is $\mathbf{y} \succeq \mathbf{y}'$ and $\exists i \in [1,\ldots,d], s.t. y_i > y'_i$.

Definition 4.3. Incomparability of vectors Objective vectors $\mathbf{y}$ and $\mathbf{y}'$ are incomparable (noted $\mathbf{y}||\mathbf{y}'$) iff $\mathbf{y} \not\succeq \mathbf{y}'$ and $\mathbf{y}' \not\succeq \mathbf{y}$.

Definition 4.4. Pareto optimality The solution $\mathbf{x}^*$ and its correspondent objective vector $\mathbf{f}(\mathbf{x}^*)$ are Pareto optimal iff $\mathbf{x} \nexists \mathscr{X}$ such that $\mathbf{f}(\mathbf{x}) \succ \mathbf{f}(\mathbf{x}^*)$.

For the sake of simplicity, we will interchangeably speak of Pareto dominance for the decision vector $\mathbf{x}$ and the associated objective $\mathbf{f}(\mathbf{x})$ in the remainder of this manuscript.

Definition 4.5. Pareto front Given a point set P, P^* is the set of points in P which are non-dominated by points in P, referred to as Pareto front w.r.t. P.

$$P^* = \{\mathbf{y} \in P : \nexists \mathbf{y}' \in P \text{ s.t. } \mathbf{y}' \succ \mathbf{y}\}$$

Definition 4.6. Optimal Pareto front P^{**} is the optimal Pareto front in the considered MOO problem if it includes all points which are non-dominated by other points in $\mathscr{X}$.

Definition 4.7. Comparison between non-dominated sets A non-dominated set P_1 is said to be better than another non-dominated set P_2 (noted $P_1 \succ P_2$) iff every $\mathbf{y} \in P_2$ is weakly dominated by at least one $\mathbf{y}' \in P_1$ and $P_1 \neq P_2$.

Definition 4.8. Pareto rank The Pareto ranks w.r.t. a set of objective vectors $P \subseteq \mathbb{R}^d$ are determined in an iterative manner as follows: all non-dominated points in P (noted as P^* or $\mathscr{F}_1(P)$) are given rank 1. The set $\mathscr{F}_1(P)$ is then removed from P; from the reduced set, the non-dominated set are given rank 2 (noted as $\mathscr{F}_2(P)$); the process continues until all points of P have received a Pareto rank. The Pareto rank of a point $p \in P$ is denoted $i_{rank}(p,P)$

Noting the largest Pareto rank of points in P by $i_{rank;max}(P)$, we have by construction $\forall i, j \in \{1,\ldots,i_{rank,max}(P)\}, i < j \Rightarrow \mathscr{F}_i(P) \succ \mathscr{F}_j(P)$.

To identify the Pareto front, there are several ways to be considered to use. In next two sections, we will address a scalarization functions like linear, Chebyshev[54] or OWA[71], which could transform the vector into scalar model; and Pareto partial order [75] allows to maximize the reward vectors directly in the multi-objective reward space.

4.1.2 Dominance method

For the MOO, the multi-objective vector could be ordered using the partial order on multi-objective spaces[75]. This method could refer to the pareto front definition.

We start with some basic definition and notation. Let $u = (u_1,\ldots,u_d) \in \mathbb{R}^d$ and $v = (v_1,\ldots,v_d) \in \mathbb{R}^d$ be two objective vectors. We define that u weakly dominates v, denoted by $u \succeq v$, precisely if $u_i \geqslant v_i$ for all $i \in [1,\ldots,d]$ and u dominates v, denoted by $u \succ v$, precisely if $u \succeq v$ and $v \not\succeq u$. We denote the set of all Boolean values by $\mathbb{B}$ and the set of all real numbers by $\mathbb{R}$ and investigate the maximization of functions fitting in the shape of $f : \mathbb{B}^n \to \mathbb{R}^d$. We call f objective function, $\mathbb{B}^n$ decision space and $\mathbb{R}^d$ objective vectors. Let $x \in \mathbb{B}^n$ and $y \in \mathbb{B}^n$ be two decision vectors. We are able to use the same manners of speaking and notations for decision vectors, since the definition $x \succeq y :\Leftrightarrow f(x) \succeq f(y)$ transfers the concept of dominance from the objective space to the decision space.

The set $\mathscr{PF}(f) := \{u \in f(\mathbb{B}^n) | \forall v \in f(\mathbb{B}^n : v \not\succ u)\}$ is called the Pareto front of f and the set $\mathscr{P}(f) := f^{-1}(\mathscr{PF}(f)) = \{x \in \mathbb{B}^n | \forall y \in \mathbb{B}^n : y \not\succ x\}$ Pareto set of f. The set $\{(x, f(x)) | x \in \mathscr{P}(f)\}$ constitutes the canonical solution of an optimization problem of the considered kind. In the literature a set of the form $\{(x, f(x) | x \in X)\}$ with $X \subseteq \mathscr{P}(f)$ is also considered as a valid solution if $f(X) = \mathscr{PF}(f)$. This means that it is sufficient to determine for all non-dominated objective vectors $u \in \mathscr{PF}(f)$ at least one decision vector $x \in \mathbb{B}^n$ with $f(x) = u$.

Global Simple Evolutionary Multi-objective Optimization (Global SEMO) [41] can be considered as one of the simplest population-based Evolutionary Algorithms for multi-objective optimization problems and has been analyzed with respect to its run-time behavior on pseudo-Boolean functions[17, 37] as well as classical combination optimization problems. Global SEMO maintains a population of variable size which serves as an archive for the discovered non-dominated individual which is drawn uniformly at random from the decision space. In each generation an individual x is drawn uniformly at random from the current population P. An offspring y is created by applying a mutation operator to X. We resort to the global to the global mutation operator which flips each bit of x with probability $1/n$ throughout this paper. The offspring is added to the population if it is not dominated by any other individual of P. All individuals which are weakly dominated by y are in turn deleted from the population. The last step ensures that the population stores for each discovered non-dominated objective vector u just

the most recently created decision vector x with $f(x) = u$. (see in Appendix A.11)

For theoretical investigations, we count the number of rounds until a desired goal has been achieved. The number of these rounds is called the runtime of the considered algorithm. The expected runtime refers to the expectation of this random variable. For exact optimization often the expected optimization time is considered which equals the expected number of iterations until a decision vector for each objective vector of $\mathscr{PF}(f)$ has been included into the population. We are mainly interested in approximations.

We are considering the following model to measure the quality of an approximation. Let $\epsilon \in \mathbb{R}^+$ be a positive real number. We define that an objective vector u ϵ-dominates v, denoted by $u \succeq_\epsilon v$, precisely if $(1+\epsilon) \cdot u_i \geqslant v_i$ for all $i \in [1,\ldots,m]$. We call a set $\mathscr{PF}_\epsilon(f) \subseteq f(\mathbb{B}^n)$ an ϵ-approximate Pareto front of f if

$$\forall u \in f(\mathbb{B}^n) : \exists v \in \mathscr{PF}_\epsilon(\boldsymbol{F}) : v \succeq_\epsilon u,$$

and a set $\mathscr{PF}^*_\epsilon(f) \subseteq \mathscr{PF}(f)$ an ϵ-Pareto front of f if $\mathscr{PF}^*_\epsilon(f)$ is an ϵ-approximate Pareto front. The corresponding Pareto sets are naturally defined, i.e., $\mathscr{P}_\epsilon(f) := f^{-1}(\mathscr{PF}_\epsilon(f))$ and $\mathscr{P}^*_\epsilon(f) := f^{-1}(\mathscr{PF}^*_\epsilon(f))$. We point out that it is possible that there are several different ϵ-approximate Pareto fronts or ϵ-Pareto fronts for a given objective function. We also emphasize that ϵ-Pareto fronts are of more value than ϵ-approximate Pareto fronts to a decision maker, since all objective vectors of an ϵ-Pareto front are non-dominated with respect to the classical concept of dominance. In the following parts, we limit our considerations to functions where the Pareto set contains all decision vectors and therefore the distinction between ϵ-approximate Pareto fronts and ϵ-Pareto fronts collapses.

Global Diversity Evolutionary Multi-objective Optimizer (Global DEMO$_\epsilon$) (shown in Appendix A.12) incorporates the concept of ϵ-dominance [41]. The idea is to partition the objective space into boxes such that all objective vectors in a box ϵ-dominate each other. The algorithm maintains at each time step at most one individual per box. This approach ensures that the individuals contained in the population show some kind of diversity with respect to their objective vectors and that the size of the population can be controlled in a better way. These properties seem to be very important if we intend to approximate a large Pareto front. We formalize this idea by introducing the so-called box index vector which maps each decision vector to the index of its box. We assume a positive and normalized objective space, i.e., $f_i(x) \geqslant 1, \forall i \in [1,\cdots,m]$ and $x \in \mathbb{B}^n$. Let

$$b_i(x) := \left\lfloor \frac{\log(f_i(x))}{\log(1+\epsilon)} \right\rfloor$$

and denote by $b(x) := (b_1(x),\ldots,b_m(x))$ the box index vector of a decision vector x. Global DEMO$_\epsilon$ works as Global SEMO with the exceptions that it does not accept an offspring with a dominated box index vector and that it deletes all individuals from the population whose box index vectors are weakly dominated by the box index vector of the offspring. This approach ensures that at most one individual per non-dominated box resides in the population.

4.1.3 Aggregation method

Except the dominance method, another useful method is construct a scalarization. A conventional way to transform a multi-objective environment into a single-objective environment is to use scalarization functions. However, since single-objective environments, in general, results in a single optimum, we need a set of scalarization functions to generate a variety of elements belonging to the Pareto optimal set. There are several types of scalarization functions that weight the values of the reward vector, but with different properties because of their (non)-linearity. We consider each set of weights to generate a scalarization function.

The linear Scalarization is the most popular scalarization function due to its simplicity. It weights each value of the reward vector and the result is the sum of these weighted values. The linear scalarized reward is

$$f(\mu_i) = \omega^1 \mu_i^1 + \cdots + \omega^d \mu_i^d, \forall i \in [1,\ldots,K]$$

where $(\omega^1,\ldots,\omega^d)$ is a set of predefined weights and $\sum_{j=1}^{d} \omega^j = 1$. A known problem with linear scalarization is its incapacity to potentially find all the points in a non-convex Pareto set.

The Chebyshev scalarization[54] has the advantage that in certain conditions it can find all the points in a non-convex Pareto set. The Chebyshev transformation was originally designed for minimization problems. The Chebyshev scalarization reward is

$$f(\mu_i) = \min_{1 \leqslant j \leqslant d} \omega^j (\mu_i^j - z^j), \forall i \in [1,\ldots,K]$$

where $z = (z^1,\ldots,z^d)$ is a reference point that is dominated by all the optimal vector μ_i^*. For each objective j, this reference point is the minimum of the current optimal minus a small positive value, $\epsilon^j > 0$. Then:

$$z^j = \min_{1 \leqslant i \leqslant d} \mu_i^j - \epsilon^j, \forall j \tag{4.2}$$

[31] shows that all the points in a Pareto set can be found by moving the reference point z.

The optimum value μ^* is the value for which the function f, linear or Chebyshev, attains its maximum value

$$f(\mu^*) := \max_{1 \leqslant i \leqslant K} f(\mu_i) \tag{4.3}$$

We denote the Pareto optimal set identifiable by the linear scalarization with A_L^* and the Chebyshev scalarization with A_C^*. The corresponding set of Pareto optimal set is O_L^* for linear scalarization and O_L^* Chebyshev scalarization.

Ordered Weighted Averaging aggregation method was proposed by Ronald R. Yager[71] in 1988. It introduces a new aggregation technique based on the Ordered Weighted Averaging(OWA) operators. OWA operators have been discussed in a large number of references.

Definition 4.9. An OWA operator of dimension n is a mapping $F : \mathbb{R}^d \to \mathbb{R}$, that has an associated n vector

$$w = (w_1, \ldots, w_d)^T$$

such as $w_i \in [0,1], 1 \leqslant i \leqslant d$, and

$$\sum_{i=1}^{d} w_i = w_1 + \cdots + w_d = 1.$$

Furthermore,

$$F(a_1, \ldots, d_d) = \sum_{j=1}^{d} w_j b_j = w_1 b_1 + \cdots + w_n b_n$$

where b_j is the j-th largest element of the bag $< a_1, \ldots, a_n >$.

A fundamental aspect of this operator is the re-ordering step, in particular an aggregate a_i is not associated with a particular weight w_i but rather a weight is associated with a particular ordered position of aggregate. When we view the OWA weights as a column vector we shall find it convenient to refer to the weights with the low indices as weight as the top and those with the higher indices with weights at the bottom.

It is noted that different OWA operators are distinguished by their weighting function. We point out three important special cases of OWA aggregations:

- **Max**: In this case $w^* = (1, 0, \ldots, 0)^T$ and

$$\mathbf{MAX}(a_1, \ldots, a_d) = \max\{a_1, \ldots, a_d\}.$$

- **Min**: In this case $w_* = (0, \ldots, 0, 1)^T$ and

$$\mathbf{MIN}(a_1, \ldots, a_d) = \min\{a_1, \ldots, a_d\}$$

- **Average**: In this case $w_A = (1/d, \ldots, 1/d)^T$ and

$$F_A(a_1, \ldots, a_d) = \frac{a_1 + \cdots + a_d}{d}$$

We can see the OWA operators have the basic properties associated with an averaging operator (commutative, monotonic and idempotent).

A window type OWA operator takes the average of the m arguments about the center. For this class of operators we have

$$w_i = \begin{cases} 0 & \text{if } i < k \\ 1/m & \text{if } k \leqslant i < k + m \\ 0 & \text{if } i \geqslant k + m \end{cases}$$

Compensative connectives have the property that a higher degree of satisfaction of one of the criteria can compensate for a lower degree of satisfaction of another criterion. Oring the criteria

means full compensation and anding the criteria means no compensation. In order to classify OWA operators in regard to their location between and and or, Yager[71] introduced a measure of orness, associated with any vector w as follows

$$\text{orness}(w) = \frac{1}{n-1}\sum_{i=1}^{n}(n-i)w_i$$

It is easy to see that for any w the orness(w) is always in the unit interval. Furthermore, note that the nearer w is to an or, the closer its measure is to one; while the nearer it is to an and, the closer is to zero. Generally, an OWA operator with much of nonzero weights near the top will be an or like operator,

$$\text{orness}(w) \geqslant 0.5$$

and when much of the weights are nonzero near the bottom, the OWA operator will be sandlike

$$\text{andness}(w) = 1 - \text{orness}(w)$$

the following theorem shows that as we move weight up the vector we increase the orness, while moving weight down causes us to decrease $orness(W)$.

Theorem 4.1. *Assume W and W' are two d-dimensional OWA vectors such that*

$$W = (w_1,\ldots,w_d)^T, W' = (w_1,\ldots,w_j+\epsilon,\ldots,w_k-\epsilon,\ldots,w_d)^T$$

where $\epsilon > 0, j < k$. Then $orness(W') > orness(W)$.

In paper[71], it defines the measure of entropy of an OWA vector by

$$\text{entropy}(w) = -\sum_{i=1}^{d} w_i \ln w_i.$$

We can see when using the OWA operator as an averaging operator $Entropy(W)$ measures the degree to which we use all the aggregates equally.

If F is an OWA aggregation with weights w_i the dual of F donated $\hat{F}$, is an OWA aggregation of the same dimension where with weights $\hat{w}_i$

$$\hat{w}_i = w_{d-i+1}$$

We can easily see that if F and $\hat{F}$ are duals then

$$Entropy(\hat{F}) = Entropy(F)$$

$$orness(\hat{F}) = 1 - orness(F) = andness(F)$$

Thus is F is or like its dual is sandlike.

An important application of the OWA operators is in the area of quantifier guided aggregations[71]. Assume

$$\{A_1, \ldots, A_d\}$$

is a collection of criteria. Let x be an object such that for any criterion A_i, $A_i(x) \in [0,1]$ indicates the degree to which this criterion is satisfied by x. If we want to find out the degree to which x satisfies "all the criteria" denoting this by $D(x)$, we get following [14].

$$D(x) = \min\{A_1(x), \ldots, A_d(x)\}$$

In this case we are essentially requiring x to satisfy A_1 and A_2 and $\ldots$ and A_d.

If we desire to find out the degree to which x satisfies "at least one of the criteria", denoting this $E(x)$, we get

$$E(x) = \max\{A_1(x), \ldots, A_d(x)\}$$

In this case we are requiring x to satisfy A_1 or A_2 or $\ldots$ or A_d.

In many applications rather than desiring that a solution satisfies one of these extreme situation, "all" or "at least one", we may require that x satisfies most or at least half of the criteria. Drawing upon [74] of linguistic quantifiers we can accomplish these kinds of quantifier guided aggregations.

Definition 4.10. A quantifier Q is called

- regular monotonically non-decreasing referred to the Picture 4.1 if

$$Q(0) = 0, Q(1) = 1, \text{ if } r_1 > r_2 \text{ then } Q(r_1) \geqslant Q(r_2).$$

- regular monotonically non-increasing referred to the Picture 4.1 if

$$Q(0) = 1, Q(1) = 0, \text{ if } r_1 < r_2 \text{ then } Q(r_1) \geqslant Q(r_2).$$

- regular unimodal referred to the Picture 4.2 if

$$Q(0) = Q(1) = 0, Q(r) = 1 \text{ for } a \leqslant r \leqslant b,$$

$$r_2 \leqslant r_1 \leqslant a \text{ then } Q(r_1) \geqslant Q(r_2), r_2 \geqslant r_1 \geqslant b \text{ then } Q(r_2) \leqslant Q(r_1).$$

With $a_i = A_i(x)$ the overall valuation of x is $F_Q(a_1, \ldots, a_d)$ where F_Q is an OWA operator. The weights associated with this quantified guided aggregation are obtained as follows

$$w_i = Q(\frac{i}{d}) - Q(\frac{i-1}{d}), i = 1, \ldots, n. \tag{4.4}$$

From the Figure 4.3 graphically shows that the operation involved in determining the OWA weights directly from the quantifier guiding the aggregation.

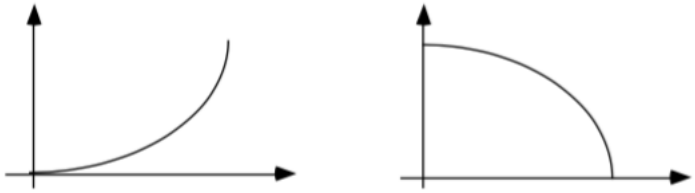

Figure 4.1: Monotone linguistic quantifiers

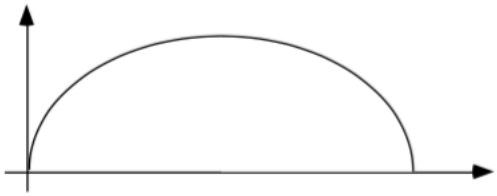

Figure 4.2: Unimodal linguistic quantifier

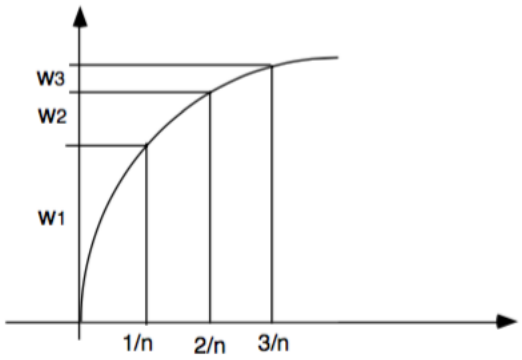

Figure 4.3: Determining weights from a quantifier.

Let us look at the weights generated from some basic types of quantifiers. The quantifier, for all Q_* referred to the figure 4.4, is defined such that

$$Q_*(r) = \begin{cases} 0 & \text{for } r < 1, \\ 1 & \text{for } r = 1. \end{cases}$$

Using this method for generating weights

$$w_i = Q_*(\frac{i}{d}) - Q_*(\frac{i-1}{d})$$

we get

$$w_i = \begin{cases} 0 & \text{for } i < n, \\ 1 & \text{for } i = n. \end{cases}$$

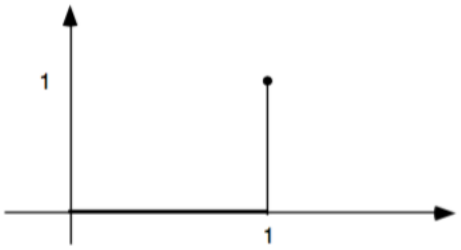

Figure 4.4: The quantifier all.

This is exactly what we previously denoted as W_*.

For the quantifier there exists (referred to the figure 4.5)we have

$$Q^*(r) = \begin{cases} 0 & \text{for } r = 0, \\ 1 & \text{for } r > 0. \end{cases}$$

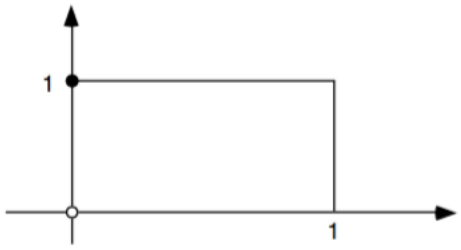

Figure 4.5: The quantifier there exists.

In this case we get

$$w_1 = 1, w_i = 0, \text{ for } i \neq 1.$$

This is exactly what we denoted as W^*.

Consider next the quantifier referred to the figure 4.6 defined by

$$Q(r) = r.$$

This is an identity or linear type quantifier.

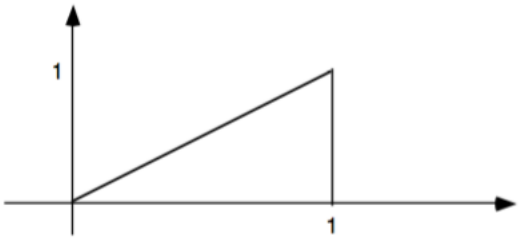

Figure 4.6: The identity quantifier.

In this case, we get

$$w_i = Q(\frac{i}{d}) - Q(\frac{i-1}{d}) = \frac{i}{d} - \frac{i-1}{d} = \frac{1}{d}.$$

This gives us the pure averaging OWA aggregation operator.

The standard degree of orness associated with a Regular Increasing Monotone (RIM) linguistic quantifier Q

$$\text{orness}(Q) = \int_0^1 Q(r)\text{d}r$$

is equal to the area under the quantifier[72]. This definition for the measure of orness of quantifier provides a simple useful method for obtaining this measure.

4.2 Multi-Objective Multi-Armed Bandit

Multi-armed bandits (MAB) is a machine learning paradigm used to study and analyze resource allocation in stochastic and uncertain environments. The multi-armed bandit problem that considers reward vectors and imports techniques from multi-obejctive optimization into the MAB algorithms is referred to as multi-objective multi-armed bandits (MOMAB).

A reward vector can be optimal in one objective and sub-optimal in the other objectives. The Pareto front contains several arms considered to be the best according to their reward vectors. Refer to the Multi-Objective Optimization problem, we could solve the MOMAB by the dominance way or construct a scalarization weight. Here, we proposed the MOMAB problem notations:

Consider an initial set of arms $\mathscr{A}$ with cardinality K, where $K \geqslant 2$ and let the vector reward space be defined as a d-dimension vector of $[0,1]^d$. When arm i is played, a random vector of reward is received, one component per objective. The random vectors have a stationary distribution with support in $[0,1]^d$. At time steps $t_1, t_2, \ldots$, the corresponding reward vectors $X_i^{t_1}, X_i^{t_2}, \ldots$ are independently and identically distributed according to an unknown law with unknown expectation vector $\mu_i = (\mu_i^1, \ldots, \mu_i^d)$. Reward values obtained from different arms are also assumed to be independent.

In the next part, we will introduce some algorithm π that chooses the strategy to find the optimal arm to play. Let $T_i(N)$ be the number of times a suboptimal arm i has been played

by π during the first N plays. The expected reward vectors are computed by averaging the empirical reward vectors observed over the time. The mean of an arm i is estimated to $\hat{\mu}_i(N) = \sum_{s=1}^{T-i(N)} X_i(s)/T_i(N)$, where $X_i(s)$ is the sampled value s for arm i.

4.2.1 Some algorithms of Multi-Objective Multi-Armed Bandit

In this section, we will introduce some algorithm to solve the MOMAB problem.

The scalarized PAC algorithm Refer to the paper[32]. In the following, it's a baseline algorithm to identify the Pareto front in stochastic environments. We assume a tolerance error δ and we want to find the Pareto front with probability at $1-\delta$. An arm i is optimal for a given scalarization function f_w iff

$$f_w(\hat{\mu}_i) > \max_{l \in \mathscr{A}} f_w(\hat{\mu}_l) - \epsilon$$

The algorithm scalarized PAC algorithm sPAC is given in Algorithm A.13. We assume a fixed number of weight vectors $W \leftarrow \{w_1, \ldots, w_{|W|}\}$. The probability that the expected reward vector of an arm i, μ_i is not the same as its estimated reward vector $\hat{\mu}_i$, is bounded with the confidence value $\epsilon > 0$ and a small error probability $\delta > 0$. We want to bound the probability of the event $f_w(\hat{\mu}_i) - f_w(\mu_i) > \epsilon$, for any scalarization function f_w.

In Algorithm A.13, each arm is pulled for an equal and fixed number of times $\frac{1}{(\epsilon/2)^2} \log(\frac{2|W|K}{\delta})$. For each weight vector w, we identify the optimal arms using the N arm pulls. The output for this algorithm is a reunion of the optimal set of arms for each scalarization function f_w. If an optimal arm is not already in the Pareto front, then that arm is added to the Pareto front $\mathscr{A}^*$. To maintain the Pareto front $\mathscr{A}*$, the dominated arms are deleted from $\mathscr{A}^*$

UCB1 in MOMAB In the MAB problems, Upper Confidence Bound(UCB) policy [8] plays firstly each arm, then adds to the estimated mean $\hat{\mu}$ of each arm i an exploration bound. The exploration bound is an upper confidence bound which depends on the number of times arm i has been selected. UCB selects the optimal arm i^* that maximizes the function $\hat{\mu}_i + \sqrt{\frac{2\ln(t)}{N_i}}$ as follows:

$$i^* = \max_{1 \leqslant i \leqslant |\mathscr{A}|} \left(\hat{\mu}_i + \sqrt{\frac{2\ln(t)}{N_i}} \right)$$

where N_i is the number of times arm i has been pulled.

For the MOMAB problems, [31] have extended the UCB policy to find the Pareto optimal arm set either by using UCB in Pareto order relationship or in scalarized functions. Where, Pareto-UCB plays initially each arm i once. At each time step t, it estimates the mean vector of each of the multi-objective arms i, i.e. $\hat{\mu}_i = [\hat{\mu}_i^1, \ldots, \hat{\mu}_i^d]$ and adds to each dimension an upper confidence bound. Pareto-UCB uses a Pareto Partial order relationships. The Pareto optimal arm set $\mathscr{A}^*$, for all the non-optimal arms k, where $k \notin \mathscr{A}^*$ there exists a Pareto optimal arm $i \in \mathscr{A}^*$

not dominates by the arms k:

$$\hat{\mu}_k + \sqrt{\frac{2\ln(t\sqrt{\lceil 4]d|\mathscr{A}^*|)}}{N_k}} \not\succ \hat{\mu}_i + \sqrt{\frac{2\ln(t\sqrt{\lceil 4]d|\mathscr{A}^*|)}}{N_i}}$$

Pareto-UCB selects uniformly, randomly one of the arms in the set $\mathscr{A}^*$. The idea is to select most of the times one of the optimal arm in the Pareto front set, $i \in \mathscr{A}^*$. An arm $j \notin \mathscr{A}^*$ that is closer to the Pareto front set according to metric measure is more selected than the arm $k \notin \mathscr{A}^*$ that is far from $\mathscr{A}^*$.

Annealing Linear Scalarized Based MOMAB algorithm This algorithm is proposed by [73]. It converts the multi-objective into a single objective one by using linear scalarized function. The annealing linear scalarized algorithm trades off efficiently between exploration and exploitation by using a decaying parameter ϵ_t, where $\epsilon_t \in (0,1)$ in combination with the Pareto dominance relation. The ϵ_t parameter has a high value at the beginning of time step t to explore all the available arms and increase the confidence in the estimated mean vectors, but as the time step t increases, the ϵ_t parameter decreases to exploit the arms that have maximum estimated mean vectors. To keep trak on all the optimal arms in the Pareto front $\mathscr{A}^*$, at each time step t, the annealing linear scalarized function uses Pareto dominance relation. If $\epsilon_t \to 0$, then the annealing algorithm selects uniformly at random one of the available arms. The beginning of time step t, ϵ_t has a high value to explore all the available arms. As the time step t is increased, ϵ_t has a low value to exploit only the optimal arms. To keep track on all the optimal arms in the Pareto front $\mathscr{A}^*$, the annealing algorithm uses Pareto dominance relation. (See in Appendix A.14)

4.3 Conclusion

Part III

Contributions

CHAPTER 5

PASSIVE-AGGRESSIVE ALGORITHM WITH BANDIT FEEDBACK

Todo.

5.1 Passive-Aggressive with Bandit feedback (PAB)

In the previous section, we have introduced the multiclass classification algorithm with Bandit feedback, i.e. Banditron, Confidit. Here, we present a novel algorithm PAB[**?**] in Algorithm 5.1, which is an adaptation of PA for the bandit case.

Similar to PA algorithm, at each round the prediction $\hat{y}_t$ is chosen by Bayesian probability according to the current weight matrix w_t, to make a reference to Eq(3.1). Unlike the conventional learning paradigm, if $\hat{y}_t \neq y_t$, it needs to perform an *exploration*, i.e sample a label randomly from [k] with parameter γ and contrast this random prediction with a bandit return $\mathbb{1}(\tilde{y}_t = y_t)$, where $\tilde{y}_t$ is the result of a random draw from a certain distribution $\mathbb{P}(\tilde{Y}|\hat{y})$.

The above intuitive argument is formalized by defining the update matrix $\tilde{U}_t$ to be a function of the random prediction $\tilde{y}_t$. We show in the following that the expectation of the PAB's update is exactly the PA's update.

PAB starts with the initiation of matrix $w_1 = \mathbf{0}$. Its update contains two items:

$$
\begin{aligned}
w_{t+1} &= w_t + U_{PAB}(x_t, \hat{y}_t, \tilde{y}_t) = w_t + U_{t,1} + U_{t,2} \\
U_{t,1} &= \frac{\mathbb{1}(\tilde{y}_t = y_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t|\hat{y}_t)} U_{PA}(x_t, \hat{y}_t, \tilde{y}_t) \\
U_{t,2} &= \frac{\mathbb{1}(\tilde{y}_t = y_t) - \mathbb{P}(\tilde{Y} = \tilde{y}_t|\hat{y}_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t|\hat{y}_t)} \cdot \rho_c \frac{\Phi(x_t, \hat{y}_t)}{2 \parallel x_t \parallel^2 + \frac{1}{2C}}
\end{aligned}
\tag{5.1}
$$

where $U_{PA}(x, \hat{y}, y)$ is the classical passive-agressive update. PAB's update contains two items. The first item is controlled by the indicator $\mathbb{1}(\tilde{y}_t = y_t)$, and is nonzero only when the true label is predicted. The role of second term is to smooth the learning process when few correct labels are available. It means that whenever the process is blind to the true label, the loss is estimated to a fixed number ρ_c; this parameter is chosen empirically.

5.1.1 Simple PAB

A simple choice is $\rho_c = 0$. The item $U_{t,1}$ is very similar to the PA's update. The following lemma is easy to prove:

Lemma 5.1. *Let $U_{t,1}$ be defined as in eq.(5.1) and let $U_{PA}(x_t, \hat{y}_t, y_t)$ be defined according to eq.(3.8). Then, $\mathbb{E}_{\tilde{Y}}[U_{t,1}] = U_{PA}(x_t, \hat{y}_t, y_t)$.*

Proof.

$$\begin{aligned}
\mathbb{E}_{\tilde{Y}}[U_{t,1}] &= \sum_{i=1}^{k} \mathbb{P}(i|\hat{y}) \cdot U_{t,1} \\
&= \sum_{i=1}^{k} \mathbb{P}(i|\hat{y}) \mathbb{1}(i = y_t) \frac{U_{PA}(x_t, \hat{y}_t, i)}{\mathbb{P}(i|\hat{y})} \\
&= \sum_{i=1}^{k} \mathbb{1}(i = y_t) U_{PA}(x_t, \hat{y}_t, i) \\
&= U_{PA}(x_t, \hat{y}_t, y_t)
\end{aligned}$$

■

By the way, simple PAB is much easy and quick to learn data, also good enough to deal with the synthetic data by the expectation U_{PA}.

5.1.2 Full PAB

Without item $U_{t,2}$, $\tilde{U}$ behaves like the U_{PA}, when $\tilde{y}_t = y_t$. And it works very well with some linear data, but it's poor to learn data real and noisy. By this appearance, we get the importance of item The second term $U_{t,2}$ is used to reduce the variance of the update. When $\rho_c > 0$, we need both $\mathbb{E}_{\tilde{Y}}[U_{t,2}] = 0$ (so that $\mathbb{E}_{\tilde{Y}}[U_{PAB}(x_t, \hat{y}_t, \tilde{Y})] = U_{PA}(x_t, \hat{y}_t, y_t)$)) and $\mathbb{E}_{\tilde{Y}}[< U_{t,1}, U_{t,2} >] \leq 0$.

Lemma 5.2. *Let $U_{t,2}$ be defined as in eq.(5.1), $\mathbb{E}_{\tilde{Y}}[U_{t,2}] = 0$.*

Proof. For each round t, we have

$$\begin{aligned}
\mathbb{E}_{\tilde{Y}}[U_{t,2}] &= \sum_{i=1}^{k} \mathbb{P}(i|\hat{y}_t) \cdot U_{t,2} \\
&= \sum_{i=1}^{k} \mathbb{P}(i|\hat{y}_t) \frac{\mathbb{I}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\rho_c}{2 \parallel x \parallel^2 + \frac{1}{2C}} \Phi(x_t, \hat{y}_t) \\
&= \sum_{i=1}^{k} (\mathbb{I}(i = y_t) - \mathbb{P}(i|\hat{y}_t)) \frac{\rho_c}{2 \parallel x \parallel^2 + \frac{1}{2C}} \Phi(x_t, \hat{y}_t) \\
&= (1 - \mathbb{P}(y_t|\hat{y})) \frac{\rho_c}{2 \parallel x \parallel^2 + \frac{1}{2C}} \Phi(x_t, \hat{y}_t) \\
&- \sum_{i \neq y_t} \mathbb{P}(i|\hat{y}_t) \frac{\rho}{2 \parallel x \parallel^2 + \frac{1}{2C}} \Phi(x_t, \hat{y}_t) \\
&= 0.
\end{aligned}$$

■

Lemma 5.3. $\mathbb{E}_{\tilde{Y}}[< U_{t,1}, U_{t,2} >] \leq 0$

Proof.

$$\begin{aligned}
\mathbb{E}_{\tilde{Y}}[< U_{t,1}, U_{t,2} >] &= \sum_{i=1}^{k} \mathbb{P}(i|\hat{y}_t) \mathbb{I}(i = y_t) \frac{U_{PA}(\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y})} \frac{\rho_c \Phi(x_t, \hat{y}_t)}{2 \parallel x \parallel^2 + \frac{1}{2C}} \\
&= \frac{1 - \mathbb{P}(y_t|\hat{y}_t)}{\mathbb{P}(y_t|\hat{y}_t)} \rho_c f(x_t, \hat{y}_t, y_t)
\end{aligned}$$

with

$$f(x_t, \hat{y}_t, y_t) = \frac{< U_{PA}(x_t, \hat{y}_t, y_t), \Phi(x_t, \hat{y}_t) >}{2 \parallel x \parallel^2 + \frac{1}{2C}}$$

then:

$$\mathbb{E}_{\tilde{Y}}[< U_{t,1}, U_{t,2} >] = \mathbb{1}(\hat{y}_t = y_t) \frac{1 - \mathbb{P}(\tilde{Y} = y_t|\hat{y}_t)}{\mathbb{P}(\tilde{Y} = y_t|\hat{y}_t)} \rho_c f(x_t, \hat{y}_t, y_t) + \mathbb{1}(\hat{y}_t \neq y_t) \frac{1 - \mathbb{P}(\tilde{Y} = y_t|\hat{y}_t)}{\mathbb{P}(\tilde{Y} = y_t|\hat{y}_t)} \rho_c f(x_t, \hat{y}_t, y_t)$$

When $\hat{y}_t = y_t$, $U_{PA} = 0$ so that $f(x_t, \hat{y}_t, y_t) = 0$. When $\hat{y}_t \neq y_t$, it can be shown that $f(x_t, \hat{y}_t, y_t) \leq 0$, so that:

$$\mathbb{E}_{\tilde{Y}}[< U_t^1, U_t^2 >] \leq 0$$

■

For an appropriate value of ρ_c, the role of $U_{t,2}$ is thus to *reduce* the variance of the PAB update, and thus improve the speed of the learning process.

Algorithm 5.1 (PAB)**.**

Initiation $w_1 = \vec{0}$.

for *for each instance* $t = 1, 2, ..., T$ ***do***

Receive $\mathbf{x}_t \in \mathbb{R}^d$

Set $\hat{y}_t = \underset{r \in \{1,...,K\}}{argmax}(W_t \Phi(x_t, r))$

$$\forall i \in [k], \mathbb{P}(\tilde{Y}_t = i | \hat{y}_t) = \mathbb{1}(\hat{y}_t = i) \cdot \epsilon + \frac{\epsilon}{k}$$

Randomly sample $\tilde{y}_t$ *according to* $\mathbb{P}(\tilde{Y}_t = i | \hat{y}_t)$

Receive the feedback $\mathbb{1}(\tilde{y}_t = y_t)$

$$l_t = < w_t, \Phi(x_t, \hat{y}_t) - \Phi(x_t, y_t) > + \mathbb{1}(\hat{y}_t = y_t)$$

$$U_{t,1} = \frac{\mathbb{1}(\tilde{y}_t = y_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)} U_{PA}(x_t, \hat{y}_t, \tilde{y}_t)$$

$$U_{t,2} = \frac{\mathbb{1}(\tilde{y}_t = y_t) - \mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)} \cdot \frac{\rho_c}{2 \parallel x_t \parallel^2 + \frac{1}{2C}} \Phi(x_t, \hat{y}_t)$$

$$U_{PAB,t}(x_t, \hat{y}_t, \tilde{y}_t) = U_{t,1} + U_{t,2}$$

Update:$W_{t+1} = W_t + U_{PAB,t}(x_t, \hat{y}_t, \tilde{y}_t)$

end for

5.1.3 Experiments

In this section, we present experimental results for the PAB and other bandit algorithms on two synthetic and one real world data sets. The cumulative loss is presented for each data set.

The first data set, denoted by SynSep, is a 9-classes, 400-dimensional synthetic data set of size 10^5. The method to generate the sample is found in [43]. The second data set, denoted by SynNonSep, is constructed in the same way as SynSep except that a 5% label noise is added, which makes the data set non-separable. The third data set is collected from the Reuters RCV1 collection. This set is made of 47236-dimensional vectors, contains 4 classes (to choose the first level categorization.) with the size of 10^5.

In the Figure 5.1 gives the cumulative loss obtained on the dataset SynSep for different online learning algorithm. Here, the Confidit algorithm (see in Appendix A.9) provides the best results, but the Simple PAB takes the second. Three out of five algorithms attain a zero loss. The worst in that case is the Banditron (see in Appendix A.8).

In the Figure 5.2, it shows the result on the SynNonSep data set, the results are rather poor in general. The Confidit and Policy Gradient [**?**] obtain the best performances, with a stable final error rate around 5%.

In the Figure 5.3, it's on the Reuters data with the first level categorization. On contrast with the synthetic datasets, the full PAB overtakes the other methods, with a final error rate around 2.5% while the other algorithms attain 5% error, and even worse in the case of Confidit (8% error rate). Besides, the PAB error rate is constantly reducing during the learning process.

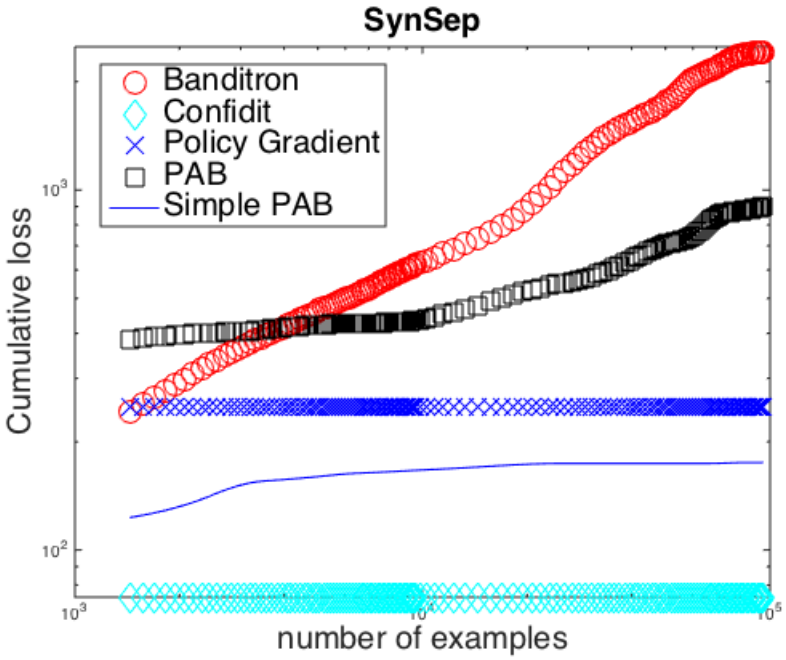

Figure 5.1: Cumulative loss of each algorithm under the SynSep data. Parameters: $\gamma = 0.014$ for Banditron, $\alpha = 1$, $\eta = 1000$ for Confidit, $\eta = 0.01$, $\lambda = 0.001$ for Policy Gradient; $C = 0.001$, $\gamma = 0.7$ and $\rho = 0$ for Simple PAB, $\rho = 1$ for Full PAB.

5.1.4 Conclusion

With the advantage of the Passive-Aggressive max-margin principle, the simple and full PAB appear effective to address the bandit online learning setting. Their first advantage is their linear complexity in space that allows to treat high dimensional datasets on the contrary to second-order methods. On separable data samples, the basic PAB overtakes most of the other approaches, at the exception of the Confidit algorithm, with a much lower complexity. It is however found to perform rather poorly on noisy and real world datasets. In contrast, the full PAB is expected to vary more smoothly over time, and is found to perform particularly well on the Reuters dataset. In that case, Confidit and Policy Gradient seem to fall in a local stable solution, while the full PAB constantly improves, issuing a better classifier. However, the performance of the algorithm is found to depend on three free parameters γ, ρ_c and C. In order to avoid fastidious cross-validation, additional investigation is needed in order to find analytic estimates of their optimal values.

5.2 Bandit feedback in Passive-Aggressive bound (BPA)

Different to the PAB algorithm, BPA [**?**] also an adaptation of PA, but its error has a same bound as PA. Similar to PAB, at each round it outputs a prediction $\hat{y}_t$ to be the label with the highest

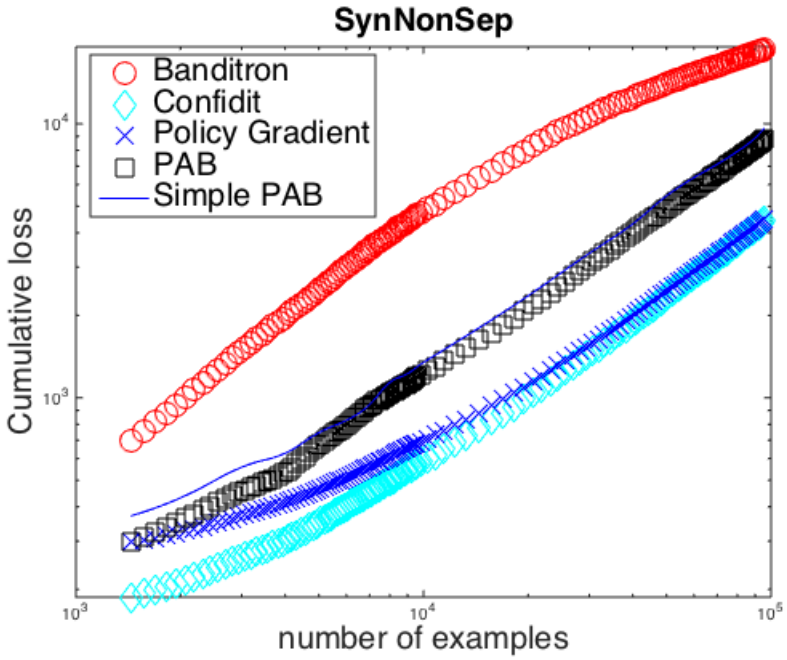

Figure 5.2: Cumulative loss of each algorithm under the SynNonSep data. Parameters: $\gamma = 0.006$ for Banditron, $\alpha = 1$, $\eta = 1000$ for Confidit, $\eta = 0.01$, $\lambda = 0.001$ for Policy Gradient; $C = 0.00001$, $\gamma = 0.7$ and $\rho = 0$ for Simple PAB, $\rho = 1$ for Full PAB.

score of $\langle w_i, x_t \rangle$, is defined as below:

$$\hat{y}_t = h_t(x_t) = \underset{i \in \{1,\dots,K\}}{\operatorname{argmax}} \langle w_i, x_t \rangle \tag{5.2}$$

where $w_i \in \mathbb{R}^d$ is the i^{th} row of the matrix $w \in \mathbb{R}^{K \times d}$. Unlike the conventional learning paradigm, if $\hat{y}_t \neq y_t$ algorithm PA to be difficult to get update because the true label's information is not supported by the bandit setting. So it need to perform an exploration, i.e., sample a label randomly $[k]$ with parameter γ and contrast this random prediction with a bandit return $\mathbb{1}_{\tilde{y}_t = y_t}$, where $\tilde{y}_t$ is the result of a random draw from a certain distribution $\mathbb{P}(\tilde{Y} | \hat{y}_t)$ by ϵ-Greedy Algorithm 2.4. Here gives an instantaneous loss by the following function,

$$l_t = [1 + (1 - 2\mathbb{1}_{\tilde{y}_t = y_t}) \langle w_{\tilde{y}_t}, x_t \rangle]_+ \tag{5.3}$$

with $(1 - 2\mathbb{1}_{\tilde{y}_t = y_t})$ equal to -1 when $\tilde{y}_t = y_t$ and 1 elsewhere. This loss is the standard hinge loss when the prediction is correct: it stays at 0 for $\langle w_{\tilde{y}_t}, x_t \rangle \geqslant 1$ and then increases for decreasing values of $\langle w_{\tilde{y}_t}, x_t \rangle$. In contrast, when the prediction is incorrect, the loss is equal to $[1 + \langle w_{\tilde{y}_t}, x_t \rangle]_+$, i.e., stays at 0 for $\langle w_{\tilde{y}_t}, x_t \rangle \leq -1$ and then increases for increasing values of $\langle w_{\tilde{y}_t}, x_t \rangle$.

The linear classifiers are updated at each trial using the standard tools from convex analysis[**?**], Where w_t satisfies the constraint in Eq. 3.6.

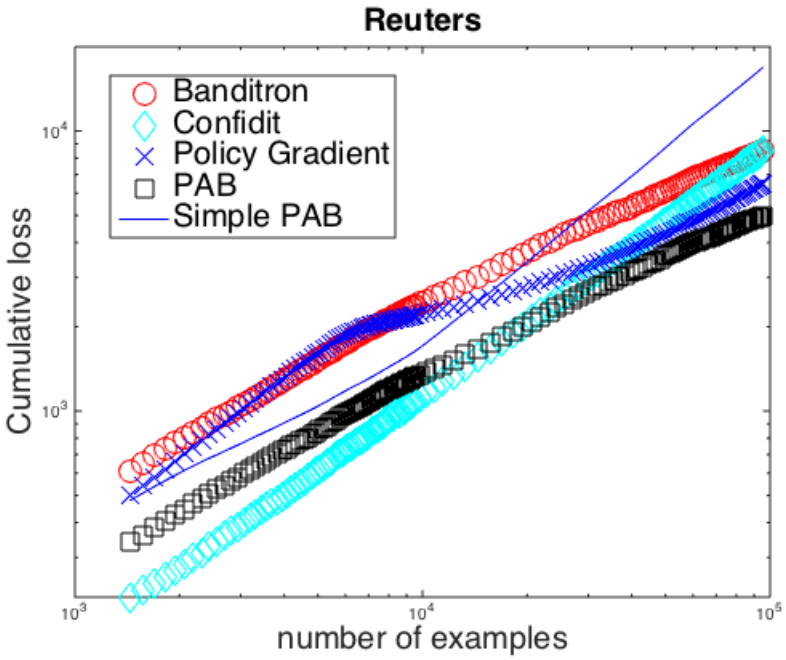

Figure 5.3: Cumulative loss of each algorithm. Parameters: $\gamma = 0.05$ for Banditron, $\alpha = 1$, $\eta = 100$ for Confidit, $\eta = 0.1$, $\lambda = 0.001$ for Policy Gradient; $C = 0.0001$, $\gamma = 0.6$ and $\rho = 0$ for Simple PAB, $\rho = 1$ for Full PAB.

$$L(w,\tau) = \frac{1}{2} \parallel w - w_t \parallel^2 + \tau \left(1 + (1 - 2\mathbb{1}_{\tilde{y}_t = y_t}) \langle w_{\tilde{y}_t}, x_t \rangle\right) \tag{5.4}$$

$$w_{t+1} = w_t + \tau \left(2\mathbb{1}_{\tilde{y}_t = y_t} - 1\right) \Phi(x_t, \tilde{y}_t) \tag{5.5}$$

Taking the derivative of $L(\tau)$ with respect to τ and also setting it to zero, we get that:

$$\tau = \frac{l_t}{\parallel \Phi(x_t, \tilde{y}_t) \parallel^2}$$

Considering for instance the common phenomenon of label noise, a mislabeled example may cause PA to drastically change its classifiers in the wrong direction. To derive soft-margin classifiers [**?**] and a non-negative slack variable ξ is introduced into the optimization problem in Equation 3.6. According with [26], the variable can be introduced in two different ways.

$$\begin{cases} w_{t+1} = \underset{w \in \mathbb{R}^{K \times d}}{argmin} \frac{1}{2} \parallel w - w_t \parallel^2 + C\xi & \text{s.t. } l(w;(x_t,y_t)) \leqslant \xi \text{ and } \xi \geqslant 0 \\ w_{t+1} = \underset{w \in \mathbb{R}^{K \times d}}{argmin} \frac{1}{2} \parallel w - w_t \parallel^2 + C\xi^2 & \text{s.t. } l(w;(x_t,y_t)) \leqslant \xi \end{cases}$$

By these optimization problems, we get the corresponding optimization solutions:

$$\begin{cases} w_{t+1} = w_t + (2\delta_{(\tilde{y}_t = y_t)} - 1) \cdot \min\left\{C, \frac{l_t}{\|\Phi(x_t, \tilde{y}_t)\|^2}\right\} \cdot \Phi(x_t, \tilde{y}_t) \\ w_{t+1} = w_t + (2\delta_{(\tilde{y}_t = y_t)} - 1) \cdot \frac{l_t}{\|\Phi(x_t, \tilde{y}_t)\|^2 + \frac{1}{2C}} \cdot \Phi(x_t, \tilde{y}_t) \end{cases}$$

Algorithm 5.2 (Bandit Passive-Aggressive)**.**

Parameter: number $\gamma \in (0,1)$.

Initialize: Set W_1 *to the zero* $K \times d$ *matrix.*

for *each round t = 1,2,. . . , n* ***do***

Observe $x_t \in \mathbb{R}^d$.

Set $\hat{y}_t = \underset{i=1,\ldots,K}{argmax} \langle W_t^i, x_t \rangle$

for all $i \in [K]$ ***do***

$\mathbb{P}(\tilde{Y} = i | \hat{y}_t) = (1-\gamma)\mathbb{1}_{i=\hat{y}_t} + \frac{\gamma}{K}$

end for

Draw $\tilde{y}_t$ *randomly from distribution* $p_t = (p_{1,t}, \ldots, p_{K,t})$.

Observe $\mathbb{1}_{(\hat{y}_t = y_t)}$.

$l_t = [\mathbb{1}_{\tilde{y}_t = y_t} + (1 - 2\mathbb{1}_{\tilde{y}_t = y_t}) \langle w_{t,\tilde{y}_t}, x_t \rangle]_+$

Update $W_{t+1} = W_t + (2\mathbb{1}_{\tilde{y}_t = y_t} - 1) \frac{l_t}{\|x_t\|^2} \cdot \Phi(x_t, \tilde{y}_t)$.

end for

5.2.1 Analysis

In this section, we prove the cumulative squared loss has a upper bound. To simplify, we note $l(w_t;(x_t,y_t))$ as l_t and $l(u;(x_t,y_t))$ as l_t^*.

Theorem 5.1. *Let* $(x_1,y_1),\ldots,(x_T,y_T)$ *be a sequence of separable examples where* $x_t \in mathbbR^d$, $y_t \in \mathscr{Y}$ *and* $\| x_t \| \leqslant R$ *for all t, and* $u \in \mathbb{R}^{K \times d}$. *Then, the cumulative squared loss of this algorithm is bounded by,*

$$\sum_{t=1}^{T} l_t^2 \leqslant R^2 \cdot \| u \|^2 \tag{5.6}$$

Proof. Define Δ_t to be:

$$\Delta_t = \| w_t - u \|^2 - \| w_{t+1} - u \|^2$$

By summing Δ_t over all t from 1 to T, that $\sum_t \Delta_t$ is a telescopic sum which collapses to,

$$\sum_{t=1}^{T} \Delta_t = \sum_{t=1}^{T} \left(\| w_t - u \|^2 - \| w_{t+1} - u \|^2 \right) = \| w_1 - u \|^2 - \| w_{t+1} - u \|^2$$

By the initiation of $w_1 = \vec{0}$,

$$\sum_{t=1}^{T} \Delta_t = \| u \|^2 - \| w_{t+1} - u \|^2 \leqslant \| u \|^2 \tag{5.7}$$

Using the definition of update in Eq.**??**,

$$\Delta_t = -2\left\langle (w_t - u), (2\delta - 1)\frac{l_t}{\| \Phi(x_t, \tilde{y}_t) \|^2}\Phi(x_t, \tilde{y}_t)\right\rangle - \left(\frac{l_t}{\| \Phi(x_t, \tilde{y}_t) \|^2}\Phi(x_t, \tilde{y}_t)\right)^2$$

With $l_t = [1 + (1 - 2\delta_{(\tilde{y}_t = y_t)}) \cdot \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$ and $l_t^* = [1 + (1 - 2\delta_{(\tilde{y}_t = y_t)}) \cdot \langle w^*, \Phi(x_t, \tilde{y}_t)\rangle]_+$, So,

$$\Delta_t = 2\frac{l_t^2 - l_t l_t^*}{\| \Phi(x_t, \tilde{y}_t) \|^2} - \left(\frac{l_t}{\| \Phi(x_t, \tilde{y}_t) \|^2} \| \Phi(x_t, \tilde{y}_t) \|\right)^2$$

$$\Delta_t = \frac{l_t^2 - 2 l_t l_t^*}{\Phi(x_t, \tilde{y}_t) \|^2}$$

If all examples are separable, $\exists u$ such that $\forall t \in [1, ..., T]$, $l_t^* = 0$, following the Eq. **??**,

$$\Rightarrow \| u \|^2 \geqslant \sum_{t=1}^{T} \Delta_t \geqslant \sum_{t=1}^{T}\left(\frac{l_t^2}{\| \Phi(x_t, \tilde{y}_t) \|^2}\right)$$

$$\Rightarrow \sum_{t=1}^{T} l_t^2 \leqslant \| u \|^2 \cdot \| \Phi(x_t, \tilde{y}_t) \|^2$$

$$\sum_{t=1}^{T} l_t^2 \leqslant R^2 \cdot \| u \|^2$$

■

Theorem 5.2. *Let* $(x_1, y_1), ..., (x_T, y_T)$ *be a sequence of non-separable examples where* $x_t \in \mathbb{R}^d$, $y_t \in [k]$ *and* $\| x_t \| \leqslant R$ *for all t. Then for any vector* $u \in \mathbb{R}^{K \times d}$ *the cumulative squared loss of this algorithm is bounded by:*

$$\sum_{t=1}^{T} l_t^2 \leqslant \left(R \| u \| + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2}\right)^2$$

Proof. By the proof of Theorem 5.1,

$$\sum_{t=1}^{T} l_t^2 \leqslant R^2 \cdot \| u \|^2 + 2\sum_{t=1}^{T} l_t l_t^*$$

To upper bound the right side of the above inequality, and denotes $L_t = \sqrt{\sum_{t=1}^{T} l_t^2}$ and $U_t = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}$,

$$2(L_t U_t)^2 - 2(\sum_{t=1}^{T} l_t l_t^*)^2 = \sum_{i=1}^{T}\sum_{j=1}^{T} l_i^2 (l_j^*)^2 + \sum_{i=1}^{T}\sum_{j=1}^{T} l_j^2 (l_i^*)^2 - 2\sum_{i=1}^{T}\sum_{j=1}^{T} l_i l_j l_i^* l_j^*$$

$$= \sum_{i=1}^{T}\sum_{j=1}^{T} (l_i l_j^* - l_j l_i^*)^2 \geqslant 0$$

$$\sum_{t=1}^{T} l_t^2 \leqslant R^2 \cdot \| u \|^2 + 2\sum_{t=1}^{T} l_t l_t^* \leqslant R^2 \cdot \| u \|^2 + 2 L_t U_t$$

$$L_t \leqslant U_t + \sqrt{R^2 \parallel u \parallel^2 + U_t^2}$$

Using the fact that $\sqrt{a+b} \leqslant \sqrt{a} + \sqrt{b}$,

$$L_t \leqslant R \parallel u \parallel + 2U_t$$

$$\sum_{t=1}^{T} l_t^2 \leqslant \left(R \parallel u \parallel + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2$$

■

5.2.2 Experiment

Here, we evaluate the algorithms over two synthetic and three real world data sets. Their characteristics are summarized in Table 5.1.

Data sets: The first data set, denoted by SynSep, is a 9-class, 400-dimensional synthetic data set of size 10^5. More details about the method to generate this data set can be found in [43]. The SynSep idea is to have a simple simulation of generating a text document. The coordinates represent different words in a small vocabulary of size 400. We ensure that SynSep is linearly separable.

The second data set, denoted by SynNonSep, is constructed the same way as SynSep except that a 5% label noise is introduced, which makes the data set non-separable.

The third data set is collected from the Reuters RCV1-v2 collection[**?**]. The original data set is composed by multi-label instances. So we make some preprocessing likes [**?**]. First, its label hierarchy is reorganized by mapping the data set to the second level of RCV1 topic hierarchy. The documents that have labels of the third or forth level only are mapped to their parent category of the second level; Second, all multi-labelled instances have been removed. This RCV1-v2 is a 53-class, 47236-dimensional real data set of size 10^5.

The fourth and fifth data sets are collected from [**? ?**]. The fourth data set is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within

Table 5.1: Summary of the five data sets, including the numbers of instances, features, labels and whether the number of examples in each class are balanced.

Data	Instances	Features	Labels	Balanced
SynSep	100 000	400	9	Y
SynNonSep	100 000	400	9	Y
RCV1-v2	100 000	47236	53	N
Letter	20 000	16	26	N
Pen-Based	13 200	16	10	N

these 20 fonts was randomly distorted to produce a file of 20000 unique stimuli. Each stimuli was converted into 16 primitive numerical attributes (statistical moments and edge counts). It forms a 26-class, 16-dimensional real data set of size 20000. The fifth data set is a digit data base made by collecting 250 samples from 44 writers, using only (x,y) coordinate information represented as constant length feature vectors, which were resampled to 8 points per digit (therefore the data set contains 8 points × 2 coordinates = 16 features). This one is a 10-class, 16-dimensional real data set of size 10992.

Results Figures 5.4 and 5.5 show the experimental results on two synthetic data sets and three real data sets. For SynSep, a separable linear data set, all algorithms except Banditron obtain a good performance; with the non-separable SynNonSep data, Confidit and BPA outperform the other algorithms, even the algorithms having a full feedback.

With the three real data sets, the algorithms with full information, despite their competitive advantage with respect to the ones with bandit feedback, do not significantly depart from BPA and Confidit, with classification results that clearly outperform Banditron. While having a lower computational complexity, BPA approach is even found to outperform Confidit in the most challenging situation, i.e. the high-dimensional case with a large number of classes (RCV1-v2 data set).

The γ parameter represents the exploration rate in Banditron and BPA algorithms. We compare on Figure 3 the average error rates obtained on the two algorithms for different values of γ on the different data sets. In contrast with Banditron, BPA shows that γ has a very little influence on the final error rate, indicating a capability to deal with small exploration rates.

5.2.3 Conclusion

In this section, we proposed a novel algorithm for online multiclass with bandit feedback. By the advantage of PA max-margin principle, BPA appears effective to address the bandit online learning setting. Its main advantage is its linear complexity in space that allows to deal with high dimensional data sets and a large number of classes, on the contrary to second-order methods. The practicability of this algorithm is verified theoretically by showing a competitive loss bound.

Moreover, experimental evaluation shows that BPA performs better than other algorithms on some real sets, even better than the algorithms with full feedback on the data sets non-separable. In the next section, we will take BPA to deal with data sets non-linear by combining the Kernel method.

5.3 Bandit feedback with kernel

5.3.1 BPA Online Kernel

For some non-linear datasets, Kernel function who transfers the data from Euclidean Space into Reproducing Kernel Hilbert Space (RKHS), has achieved great success in various problems

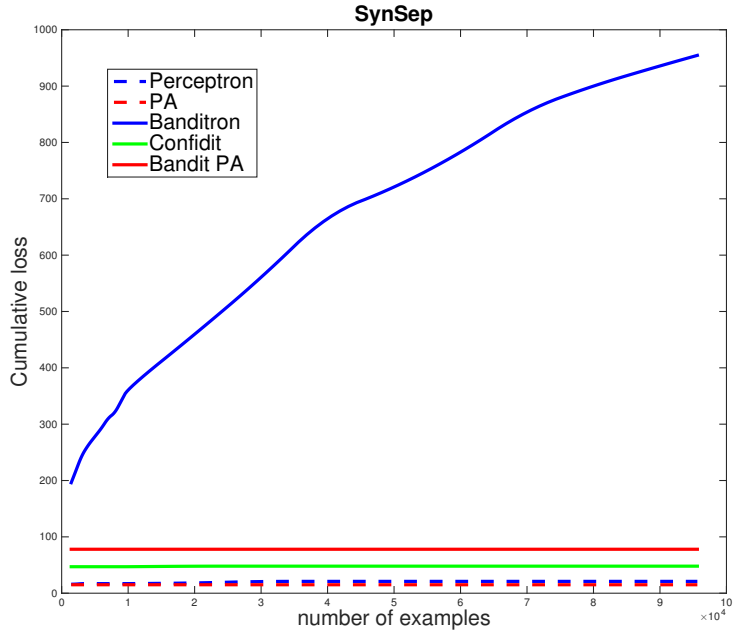

Figure 5.4: Cumulative Errors on the synthetic data set of SynSep.

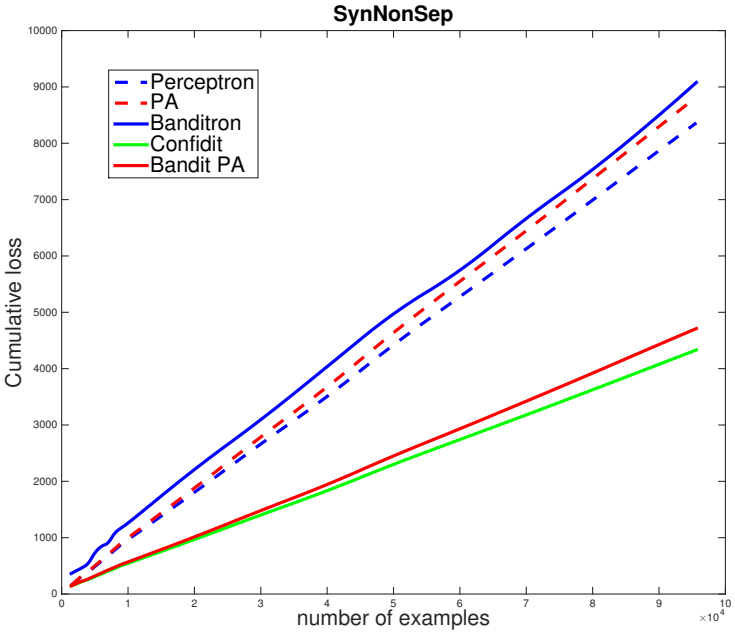

Figure 5.5: Cumulative Errors on the synthetic data set of SynNonSep.

where all of the training data could be observed in advance. And some kernel based algorithms, i.e. SVM, exhibit extraordinary performance. In recent years, this issue has been studied by more

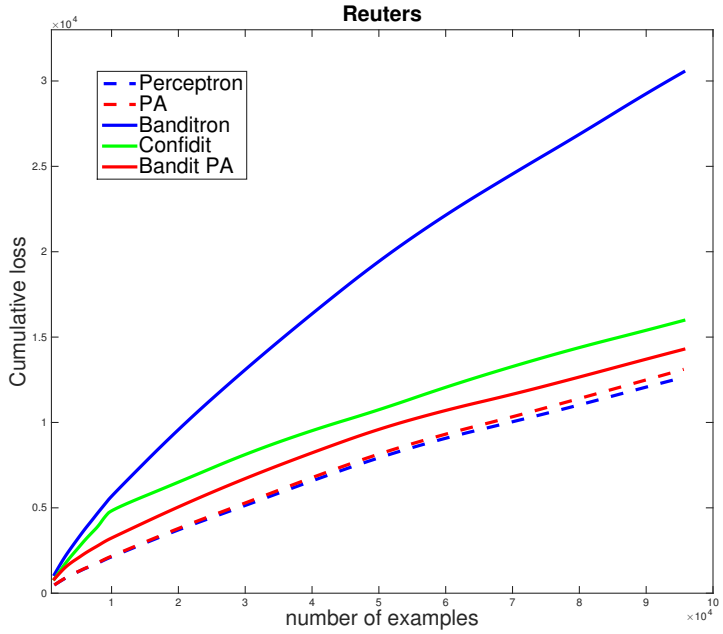

Figure 5.6: Cumulative Errors on the real data set of RCV1-v2 (53 classes).

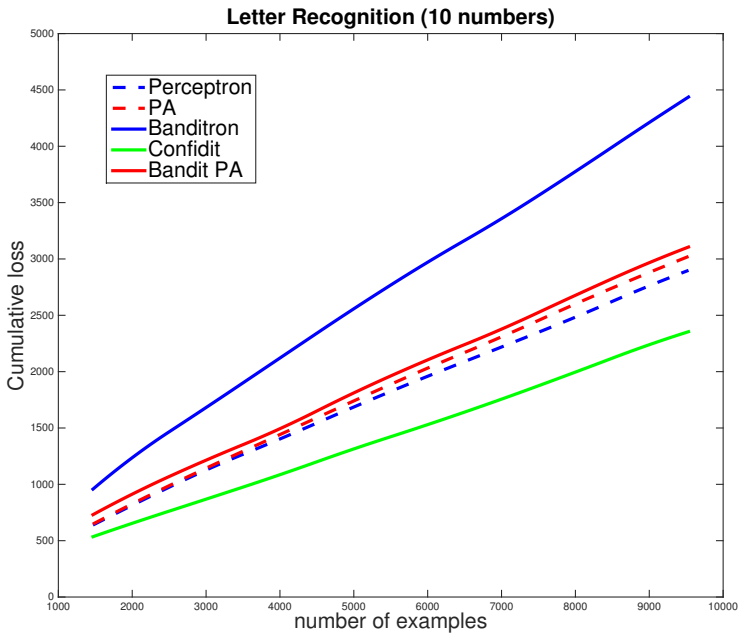

Figure 5.7: Cumulative Errors on the real data set of Letter Recognition (10 numbers).

and more researcher. In [**? ? ?**], they focus on another kind of kernel framework, the kernel in an online setting suitable for real-time application. They propose the method to take the online

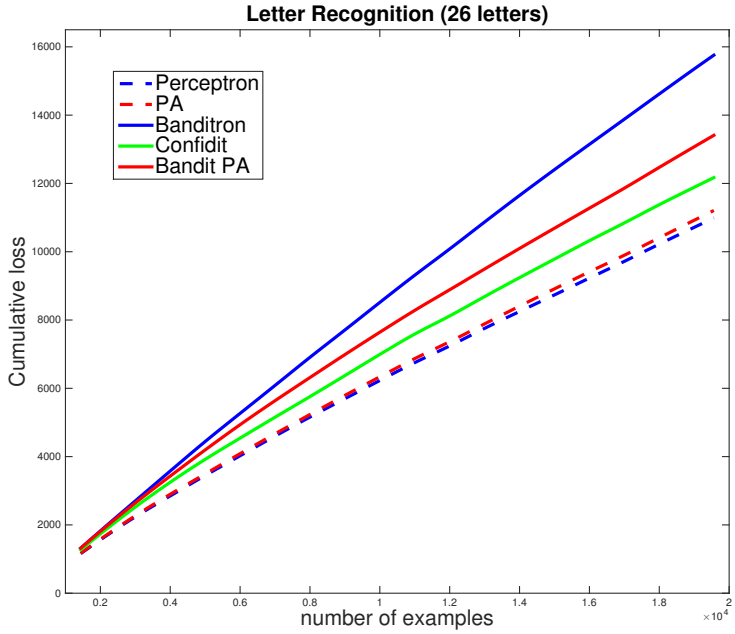

Figure 5.8: Cumulative Errors on the real data set of Letter Recognition (26 Letters).

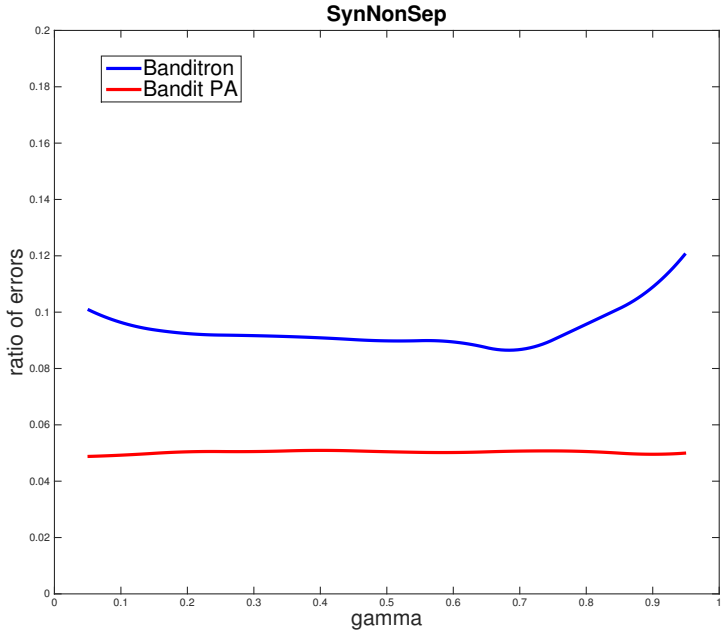

Figure 5.9: Average error of Banditron and BPA for parameter's value γ on the data set of SynNonSep.

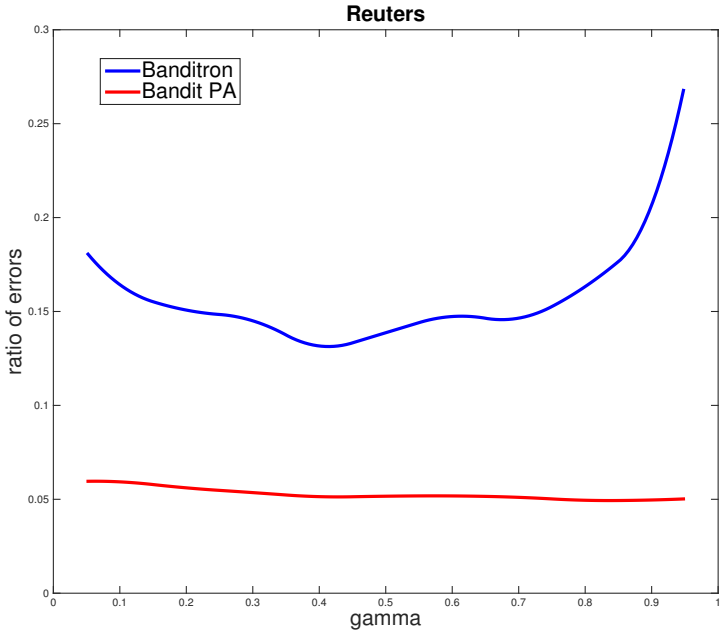

Figure 5.10: Average error of Banditron and BPA for parameter's value γ on the data set of Reuters.

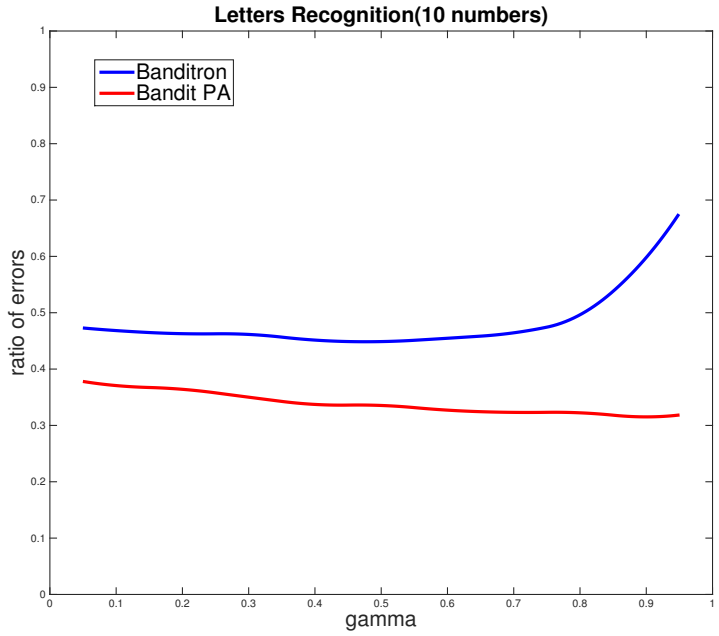

Figure 5.11: Average error of Banditron and BPA for parameter's value γ on the data set of Letter Recognition.

learning in a Reproducing Kernel Hilbert Space, by considering the classical stochastic gradient descent.

In this section, we proposed two kernel based algorithm to solve the problem of online learning with partial feedback. Before to introduce the method, we present some notions will be used later. The goal of online learning is to output a set of classifier $\mathscr{F} = [f^1, \ldots, f^K]$, from the sets of all hypothesis $\mathscr{H}$, where $f^i : \mathscr{X} \to \mathbb{R}$.

We assume that $\mathscr{H}$ is a Reproducing Kernel Hilbert Space. It means there exists a kernel function $k : \mathscr{X} \times \mathscr{X} \leftrightarrow \mathbb{R}$ and a dot product $< \cdot, \cdot >_{\mathscr{H}}$ such that

- $k(x_1, x_2) =< x_1, x_2 >$
- $< f, k(x, \cdot) >_{\mathscr{H}} = f(x)$, for $x \in \mathscr{X}$
- $\| f \|_{\mathscr{H}} =< f, f >_{\mathscr{H}}^{\frac{1}{2}}$

.

Refer to the definitions of BPA algorithm, at each round, the output $\hat{y}_t$ is to be predicted by the Function 3.1. To adapt to the environment of RKHS:

$$\hat{y}_t = \arg\max_{i \in [K]} f^i(x)$$

.

Considering the update of BPA algorithm Function 5.5, its update for RKHS will be shown as below:

$$f_{t+1}^{\tilde{y}_t} = f_t^{\tilde{y}_t} + \tau_t \cdot (2\mathbb{1}_{(\tilde{y}_t = y_t)} - 1) \cdot k(\mathbf{x}_t, \cdot)$$

where, $\tau_t = \frac{l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)})}{k(\mathbf{x}_t, \mathbf{x}_t)}$ with the loss function

$$l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)}) = [\mathbb{1}_{(\tilde{y}_t \neq y_t)} + (1 - 2\mathbb{1}_{(\tilde{y}_t = y_t)}) f_t^{\tilde{y}_t}(\mathbf{x}_t)]_+$$

.

So that, following the growth of learning examples, we will get the hypothesis $\mathscr{F}$:

$$\forall k \in \{1, \ldots, K\}, f_t^k = \sum_{i=1}^{t-1} \mathbb{1}(k = \tilde{y}_i) \tau_i \cdot (2\mathbb{1}(\tilde{y}_i = y_i) - 1) \cdot k(x_i, \cdot) \tag{5.8}$$

Algorithm 5.3 (The BPA algorithm in RKHS)**.**

A sequence of learning data $x_1, \ldots, x_T$

Creat K *containers* $\mathscr{C} = \{C_1, \ldots, C_K\}$*, each one has a zero vector* $\mathbf{0} \in \mathbb{R}^d$ *as initialization.*

for *On each round* $t \in \{1, 2, \ldots, \}$ ***do***

Receive the instance data $\mathbf{x}_t$

Predict $\hat{y}_t = \underset{i \in \{1, \ldots, K\}}{argmax} \sum_{s=1}^{|\mathscr{C}|} \alpha_s^i f_s^i(x_t)$ *where* f_s^i *is the* s^{th} *support vector from* C_i *container.*

for *all* $i \in \{1, \ldots, K\}$ ***do***

$\mathbb{P}(\tilde{Y} = i | \hat{y}_t) = (1 - \epsilon)\mathbb{1}_{i = \hat{y}_t} + \frac{\epsilon}{K}$

end for

Draw $\tilde{y}_t$ *randomly from the distribution* $p_t = (p_{1,t}, \dots, p_{K,t})$

Observe the Bandit feedback $BF = \mathbb{1}_{\tilde{y}_t = y_t}$

$l_t = [\mathbb{1}_{\tilde{y}_t \neq y_t} + (1 - 2\mathbb{1}_{\tilde{y}_t = y_t}) \sum_{s=1}^{|\mathscr{C}|} \alpha_s^{\tilde{y}_t} f_s^{\tilde{y}_t}(x_t)]_+$

If $l_t \neq 0$*, the* $C_{\tilde{y}_t}$ *container should take a new support vector* $\mathbf{x}_t$ *with its coeffcient* $\alpha_{|\mathscr{C}+1|}^{\tilde{y}_t}$.

end for

By the rule of update, it clearly shows that the hypothesis is composed by limited support vectors if the data is separable. If not, the number of support vectors grows without bounds. In that case, we can use another way to solve this issue in the next part.

5.3.2 Online Stochastic Gradient Descent Kernel with BPA loss

This method should refer to Kivinen's [**?**] Stochastic Gradient Descent. Like the SGD in Hilbert Space, the goal of SGD is to minimize the regularized risk:

$$R[\mathscr{F}] = \mathbb{E}[l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)})] + \frac{\lambda}{2} \parallel \mathscr{F} \parallel_{\mathscr{H}}^2$$

Here, the loss function should be replaced by the function 5.3 the loss function of algorithm BPA. Consider the classical stochastic gradient descent, take the gradient gradient to each hypothesis f^i of $\mathscr{F}$.

$$\forall k \in [K], f_{t+1}^k = f_t^k - \eta_t \partial_{f^k} R[\mathscr{F}]|_{f^k = f_t^k} \tag{5.9}$$

where for $k \in [K]$, $t \in \mathbb{N}$, $f_t^k \in \mathscr{H}$, ∂_{f^k} denote $\partial / \partial f^k$ and $\eta_t > 0$ is the learning rate (in this section, it is considered as a constant $\eta_t = \eta$).

Since,

$$\begin{aligned} \partial_{f^k} R[\mathscr{F}] &= \frac{\lambda}{2} \partial_{f^k} \parallel \mathscr{F} \parallel_{\mathscr{H}}^2 + \partial_{f^k}(\mathbb{E}[l(x_t, \tilde{y}_t)]) \\ &= 2f^k + \partial_{f^k} l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)}) \end{aligned}$$

$$\partial_{f^k} l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)}) = \begin{cases} 1 - 2\mathbb{1}_{(\tilde{y}_t = y_t)} & \text{if } k = \tilde{y}_t \\ 0 & \text{else} \end{cases}$$

$$f_{t+1}^k = \begin{cases} f_t^k \cdot (1 - \lambda\eta) + \eta \cdot (2\mathbb{1}_{(\tilde{y}_t = y_t)} - 1) \cdot k(x_i, \cdot) & \text{if } k = \tilde{y}_t \\ f_t^k & \text{else} \end{cases} \tag{5.10}$$

Here, we propose some ordered parameters $(\sigma_t^1, \dots, \sigma_t^K)$ with

$$\sigma_t^k = \sum_{s=1}^{t} \mathbb{1}(\tilde{y}_s = k) \tag{5.11}$$

By the parameters σ, the update function 5.10 could be expressed as the following equation: for $\forall k \in \{1,\dots,K\}$

$$f_{t+1}^k = \sum_{i=1}^{t} \eta \alpha_i^k \cdot k(x_i, \cdot) \tag{5.12}$$

where $\alpha_i^k = \mathbb{1}_{(k=\tilde{y}_i)}(2\mathbb{1}(\tilde{y}_i = y_i) - 1) \cdot (1-\lambda\eta)^{\sigma_t^k - \sigma_i^k - 1}$.

Consider that the update 5.12 contains σ_t kernel expansion terms, since the amount of computation terms grows linearly in the size of the expansion as same as the method we just mentioned in the head of this section. We learn the way from [**?**] proposed by Kivinen, at each iteration the coefficients α_i^k are shrunk by $1-\lambda\eta$ except $i = t$. Thus after τ iterations the coefficients α_i^k will be reduced to $(1-\lambda\eta)^\tau$. Hence it could drop small terms and incur little error as the following lemma.

Lemma 5.4. ***Truncation Error***

Suppose $l_t(\mathbf{x}_t, \mathbb{1}(\tilde{y}_t = y_t))$ is the loss function which satisfies the condition $|\partial_{f^k} l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)})| \leqslant C$ for all $k \in [K]$ and $k(\cdot,\cdot)$ is a kernel function with bounded norm $\| k(x,\cdot) \| \leqslant X$ where $\|\cdot\|$ denotes $\|\cdot\|_{\mathscr{H}}$. Let $f_{trunc}^k = \sum_{i=max(1,t-\sigma_{i'}^k)}^{t} \eta \alpha_i k(x_i,\cdot)$ denote the kernel expansion truncated to $\sum_{s=\sigma_{i'}^k}^{t} \mathbb{1}(\tilde{y}_s = k) = \tau$ terms. The truncation error satisfies that for each $k \in [K]$

$$\| f^k - f_{trunc}^k \| \leqslant \sum_{i=1}^{t-\sigma_{i'}^k} \eta(1-\lambda\eta)^{t-\sigma_i^k} CX < (1-\lambda\eta)^\tau CX/\lambda$$

Obviously the approximation quality increases exponentially with the number of terms retained.

Algorithm 5.4 (SGD with BPA loss in RKHS)**.**

A sequence of learning data $x_1,\dots,x_T$
Initialize the parameters $\lambda > 0$, a truncation parameter $\tau \in \mathbb{N}$, a learning rate $\eta \in (0, 1/\lambda)$
for** On each round $t \in \{1,2,\dots,\}$ **do
  Receive the instance data x_t
  Predict $\hat{y}_t = \underset{i\in\{1,\dots,K\}}{argmax}\ f_t^i(x_t)$
  for** all $i \in \{1,\dots,K\}$ **do
    $\mathbb{P}(\tilde{Y} = i | \hat{y}_t) = (1-\epsilon)\mathbb{1}_{i=\hat{y}_t} + \frac{\epsilon}{K}$
  end for
  Draw $\tilde{y}_t$ randomly from the distribution $p_t = (p_{1,t},\dots,p_{K,t})$
  Observe $\mathbb{1}_{\tilde{y}_t = y_t}$
  $l_t = [\mathbb{1}_{\tilde{y}_t \neq y_t} + (1 - 2\mathbb{1}_{\tilde{y}_t = y_t}) f_t^{\tilde{y}_t}(x_t)]_+$
  $f_{t+1}^{\tilde{y}_t} = \sum_{s=max(0,\tau)}^{t} \eta \alpha_s^{\tilde{y}_t} \cdot k(x_i,\cdot)$
  where $\alpha_s^{\tilde{y}_t} = \mathbb{1}_{k=\tilde{y}_s}(2\mathbb{1}_{\tilde{y}_s = y_s} - 1) \cdot (1-\lambda\eta)^{\sigma_t^{\tilde{y}_s} - \sigma_s^{\tilde{y}_s} - 1}$ and $\sigma_t^k = \sum_{s=1}^{t} \mathbb{1}(\tilde{y}_s = k)$
end for

5.3.3 Experimentation

In this section, we take two datasets to evaluate and analyze the effect of these algorithm in Reproducing Kernel Hilbert Space.

Data description The first dataset denoted by Pendigits, is a real data and created by E.Alpaydin and Fevzi.Alimoglu [**? ?**]. It collected 250samples from 44 writers. All writers are asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution. Here, the dataset is part of original one. It contains 7494 instances, 16 features and 10 classes.

The second dataset denoted by 'Segment'[**?**]. This dataset contains 2310 instances, all of them were drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a clasification for every pixel. Each instance is a 3×3 region. It's a real dataset, with 19 features and 7 classes. More details could be referred to the data site "UCI".

Algorithm Here, we take two algorithms KBPA and KSGD to compare. In order to perform the effect of kernel, we choose KBPA in linear model for comparison. So, all participant algorithms contains: KBPA(linear), KBPA(Laplace), and KSGD. For each dataset, the parameter of kernel function is different. By cross-validation way, we choose $\eta = 1$ of model 'Laplace' for dataset Pendigits and $\eta = 10$ for dataset 'Segment'. For KSGD, the truncated number is 500 for dataset Pendigits, and 200 for Segment.

Result

Here, it should present more details of the result,

- pendigits non separable, so for KBPA(laplace), the kernel always augments, it has better precision, but the training time is multiple of others.
- Segment is a separable dataset, so the training time is similar between KSGD and KBPA, but KBPA is still has a better performance than KSGD.
- Why they have the same Regret?

5.3.4 Conclusion

5.4 Passive-Aggressive algorithm with Bandit feedback for multilabel classification

Before introduce this algorithm, let's get some notations and preliminaries. It is applied in a sequence of consecutive rounds. On round t, the learner is given an instance vector $\mathbf{x}_t \in \mathscr{X}$ and outputs a binary vector $\hat{Y}_t \in \{0,1\}^K$ representing a label subset from the set of all labels. In the general setting, the true response $Y_t \in \{0,1\}^K$ associated with $\mathbf{x}_t$ is generated after the prediction. In the bandit setting, the learner can not observe Y_t, only receive $\beta_t \in \{0, \frac{1}{2}, 1\}^K$ as partial feedback,

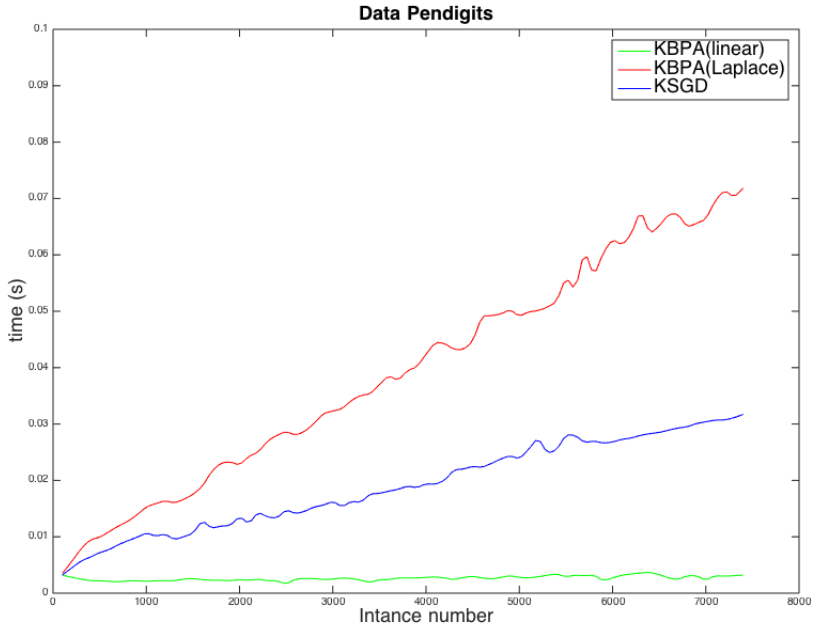

Figure 5.12: Average training time for each instance of Data Pendigits.

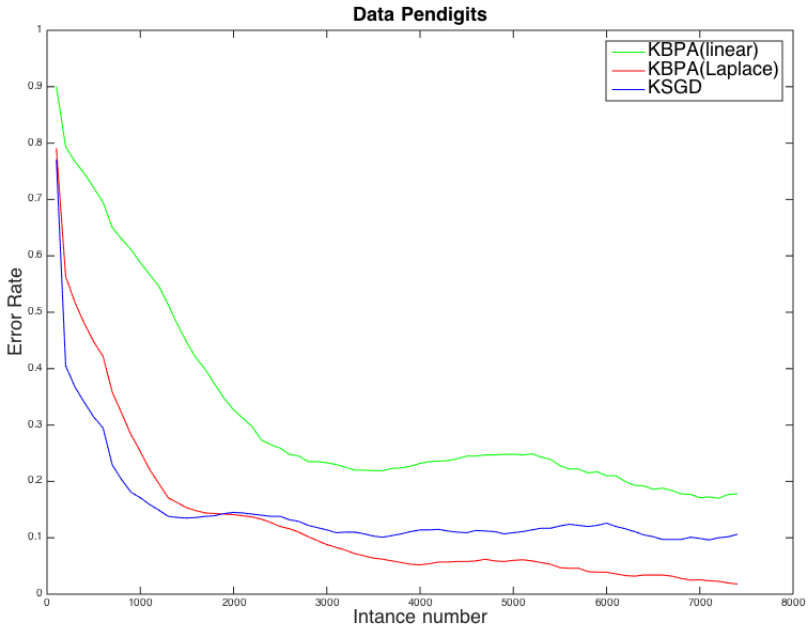

Figure 5.13: Average error rate for each instance of Data Pendigits

where for $\forall k \in \{1,\ldots,K\}$:

$$\beta_t^k = \begin{cases} 1 & \hat{y}_t^k = y_t^k = 1 \\ 0 & \hat{y}_t^k = 1 \& y_t^k = 0 \\ \frac{1}{2} & \hat{y}_t^k = 0 \end{cases} \tag{5.13}$$

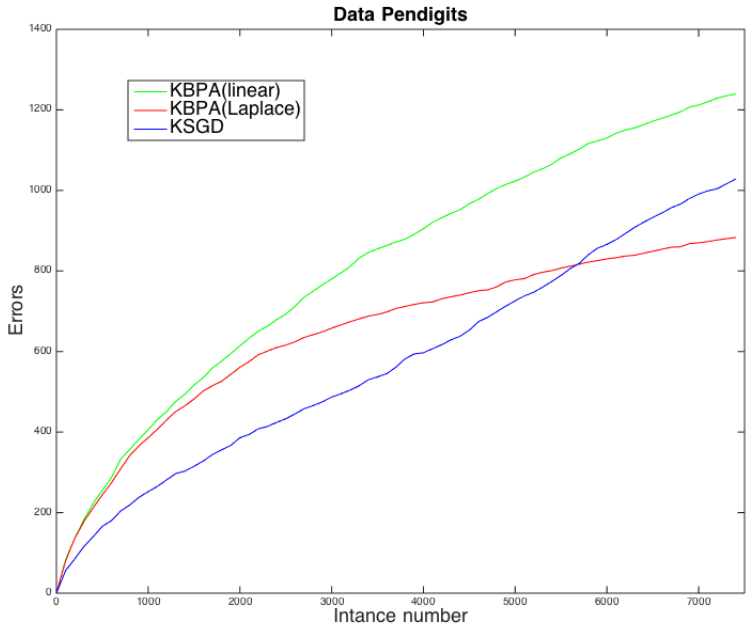

Figure 5.14: Cumulative Errors of Data Pendigits

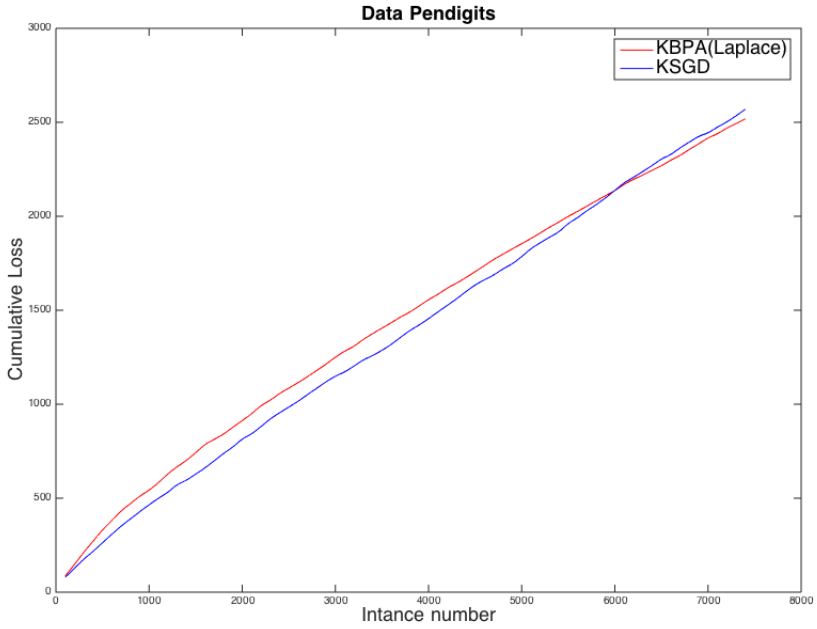

Figure 5.15: Cumulative loss of Data Pendigits

The widely known Binary relevance Method(BM) [**?**] transfers the multi-label task into several single independent binary classification tasks for each label. BM approach is theoretically simple

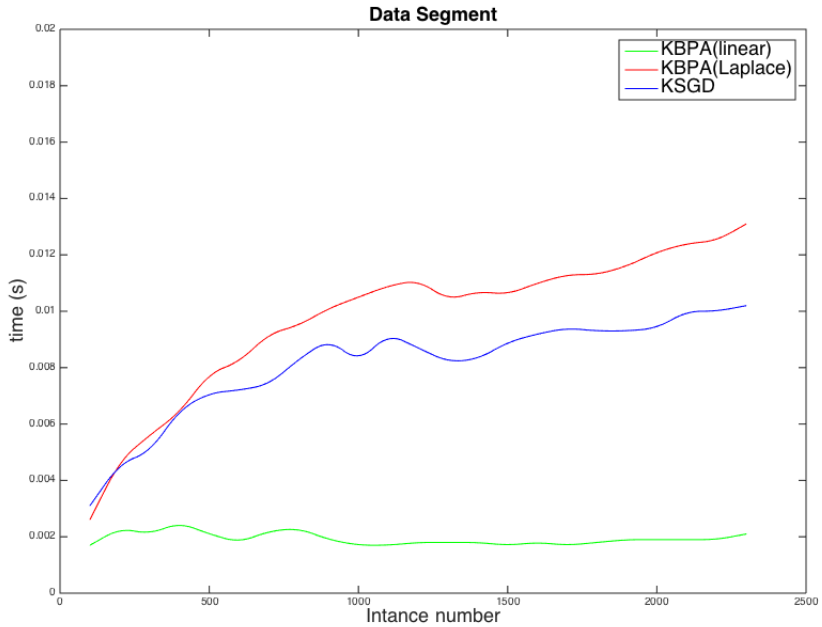

Figure 5.16: Average training time for each instance of Data Segment.

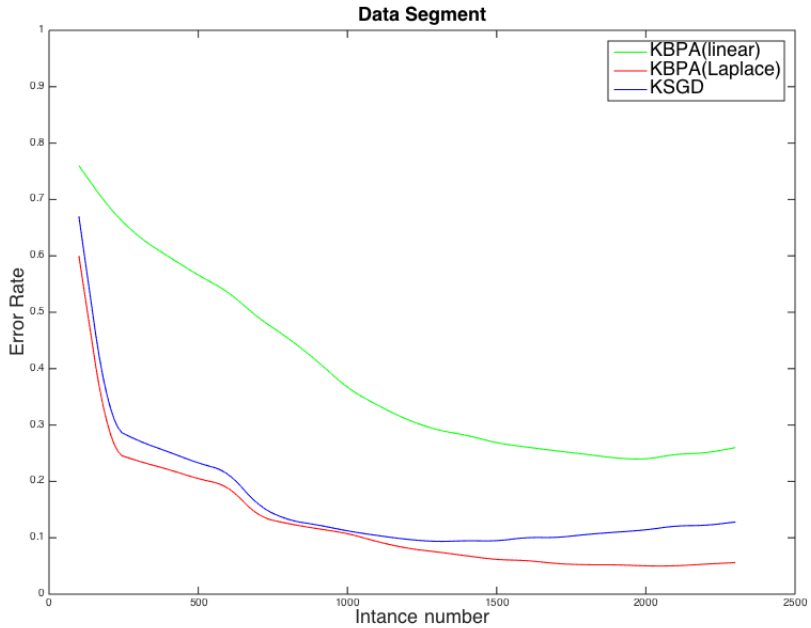

Figure 5.17: Average error rate for each instance of Data Segment

and intuitive. Its assumption of label independence makes it ignores label correlations that exist in the training data. The most important advantage of BM is its low computational complexity

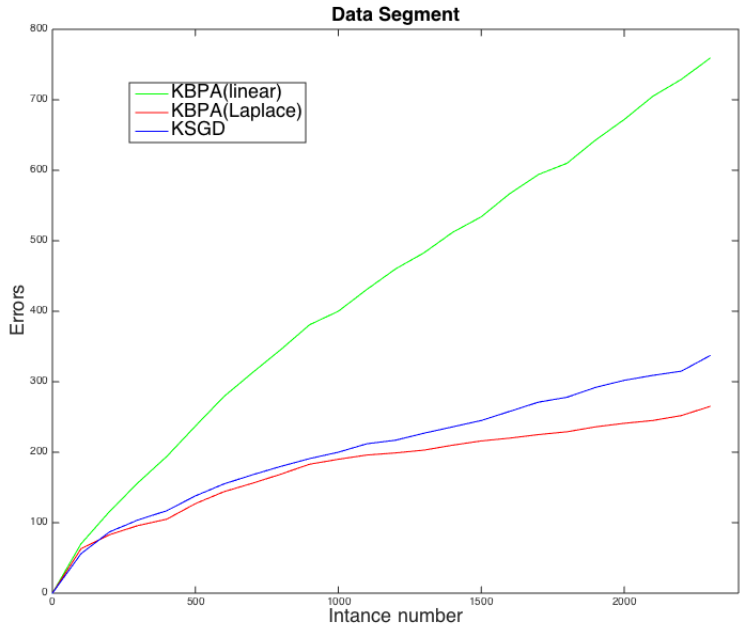

Figure 5.18: Cumulative Errors of Data Segment

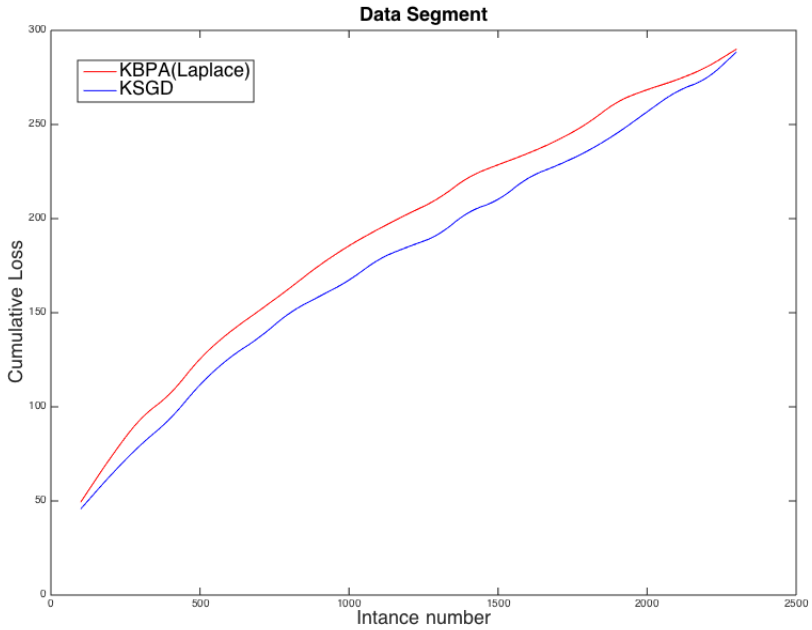

Figure 5.19: Cumulative loss of Data Segment

compared to other methods.

We consider a multilabel classification model $\mathbf{f}(\mathbf{x})$ is a mapping function $\mathscr{X} \rightarrow \{0,1\}^K$. The

output of the model is a binary vector:

$$\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_K(\mathbf{x}))$$

here $\mathbf{h}(\mathbf{x})$ is taken from a class of hypothesis $\mathbf{H}$ parameterized by a $K \times d$ matrix of real weights W. Hence, $\forall i \in [K].h_i() := \frac{1}{2}(1 + sign < W^i, x >)$ where W^i is the i^{th} row of W. If the score $< W^i, x >$ is positive, the prediction is $\hat{y}_t^i = 1$, else it equals 0.

The algorithm BPA is based on the online Passive-Aggressive approach. Consistently with paper [26]'s writing, a feature function: $\Phi(x, i)$ is a $K \times d$ matrix which is composed of K features vectors of size d. All rows of $\Phi(x, i)$ are zero except the i^{th} row which is set to x. It can be remarked that $< \Phi(x, i), \Phi(x, j) >= \| x \|^2$ if $i = j$ and 0 otherwise. In our case, $< W, \Phi(x, i) >$ is equal to $< W^i, x >$.

In the bandit setting, a strategy needs to be set to address the exploration/exploitation trade off. In this paper, we use the strategy ϵ-greedy in Section **??**. At each round $t = 1, 2, \ldots$ the algorithm selects a subset $\tilde{Y}_t$ according to the probabilities $P_t = \left(P(\tilde{y}_t^1) = 1 | \hat{y}_t^1, \ldots, P(\tilde{y}_t^K = 1 | \hat{y}_t^K)\right)$, with:

$$p(\tilde{y}_t^i = 1 | \hat{y}_t^i) = (1 - \epsilon) \cdot \mathbb{1}_{(\hat{y}_t^i = 1)} + \epsilon \cdot \frac{\sum_{k=1}^K \mathbb{1}_{(\hat{y}_t^k = 1)}}{K} \tag{5.14}$$

Let the cardinality of $\hat{Y}$ be noted $Card(\hat{Y}) = \sum_{k=1}^K \mathbb{1}_{(\hat{y}^k = 1)}$.

Lemma 5.5. *With the notations introduced so far, when following an ϵ-greedy random selection, the expected cardinality of $\tilde{Y}_t$ is equal to $Card(\hat{Y}_t)$.*

Proof.

$$\mathbb{E}[Card(\tilde{Y}_t)] = \sum_{k=1}^K P(\tilde{y}_t^k = 1 | \hat{y}_t^k)$$

$$\mathbb{E}[Card(\hat{Y}_t)] = (1 - \epsilon) \sum_{k=1}^K \mathbb{1}_{\hat{y}_t^k = 1} + \epsilon \sum_{k=1}^K \mathbb{1}_{\hat{y}_t^k = 1}$$

$$\mathbb{E}[Card(\tilde{Y}_t)] = \sum_{k=1}^K \mathbb{1}_{\hat{y}_t^k = 1} = Card(\hat{Y}_t)$$

■

The instantaneous loss is defined as a piece-wise hinge loss function $L_t = \sum_{k=1}^K l_t^k$, where

$$l_t^k = [\mathbb{1}_{\tilde{y}_t^k = 1} + \left(1 - 2\beta_t^k\right) \langle W, \Phi(x_t, k) \rangle]_+ \tag{5.15}$$

with $1 - 2\beta_t^i$ equals to -1 when $\tilde{y}_t^i = y_t^i = 1$, $+1$ when $\tilde{y}_t^i = 1$ & $y_t^i = 0$ and 0 otherwise (see Algorithm 5.5). This loss is the standard hinge loss $[1 - \langle W, \Phi(x_t, \tilde{y}_t) \rangle]_+$ when the prediction is positive correct: it stays at 0 for $\langle W, \Phi(x_t, \tilde{y}_t) \rangle \geqslant 1$ and then increases for decreasing values of $\langle W, \Phi(x_t, \tilde{y}_t) \rangle$. In contrast, when the prediction is positive incorrect, the loss is equal to $[1 +$

$\langle W, \Phi(x_t, \tilde{y}_t)\rangle]_+$, i.e., stays at 0 for $\langle W, \Phi(x_t, \tilde{y}_t)\rangle \leq 1$ and then increases for increasing values of $\langle W, \Phi(x_t, \tilde{y}_t)\rangle$. For the negative prediction $\tilde{y}_t^i = 0$, the loss equals to 0.

In this section, we introduce a novel online learning algorithm (see algorithm 5.5), which is a variant of the Online Passive-Aggressive algorithm adapted to the multi-label classification in the bandit setting. It is named BPAs (Bandit Passive-Aggressive with multiple labels).

The goal of online learning is to minimize the cumulative loss for a certain prediction task from sequentially arriving training samples. Online Passive-Aggressive achieves this goal by updating some parameterized model W in an online manner with the instantaneous losses from the arriving data x_t and corresponding responses y_t, with $t \geqslant 0$. The loss $l(W;(x_t, y_t))$ can be the hinge loss. The solution will be derived from the optimization problem:

$$\begin{aligned} W_{t+1} &= \underset{W \in \mathbb{R}^{K \times d}}{argmin} \frac{1}{2} \parallel W - W_t \parallel^2 \\ &s.t.\ l_t(W;(x_t, y_t)) = 0 \end{aligned} \tag{5.16}$$

Intuitively, if W_t suffers no loss from new data, i.e., $l(W;(x_t, y_t)) = 0$, the algorithm passively assigns $W_{t+1} = W_t$; otherwise, it aggressively projects W_t to the feasible zone of parameter vectors that attain zero loss.

Similarly to Online Passive-Aggressive, we output at each round a subset prediction $\hat{Y}_t \in \{0,1\}^K$ according to the binary prediction as following:

$$\hat{y}_t^i = \text{sign}(< W, \Phi(x, i) >)$$

Then an ϵ-greedy exploration is performed, where $\tilde{Y}_t \in \{0,1\}^K$ is the result of a random draw from $\mathbf{P}_t$ (see Eq.(5.14)).

The linear classifiers are updated at each trial using the standard tools from convex analysis [**?**]. According the instantaneous loss (see Eq.(5.15)), the constrained optimization problem of Eq.(5.16) is solved by the Lagrangian method, i.e.:

$$\mathscr{L}(W, \tau_t) = \frac{1}{2} \parallel W - W_t \parallel^2 + \tau_t \sum_{k=1}^{K} l_t^k \tag{5.17}$$

$$\frac{\partial \mathscr{L}(W, \tau_t)}{\partial W} = W - W_t + \tau_t \sum_{k=1}^{K} (1 - 2\beta_t^k)\Phi(x_t, k)$$

$$\Rightarrow W = W_t + \tau_t \sum_{k=1}^{K} \left(2\beta_t^k - 1\right)\Phi(x_t, k) \tag{5.18}$$

$$\frac{\partial \mathscr{L}(\tau_t)}{\partial \tau_t} = 0$$

$$\Rightarrow \tau_t = \frac{L_t}{(\sum_{k=1}^{K}(2\beta_t^k - 1)\Phi(x_t, k))^2}$$

$$\tau_t = \frac{L_t}{\sum_{k=1}^{K}(2\beta_t^k - 1)^2 \parallel x_t \parallel^2} \tag{5.19}$$

Consider for instance the common phenomenon of label noise, a mislabeled example may cause PA to drastically change its classifiers in the wrong direction. To derive soft-margin (Vapnik, 1998), a non-negative slack variable ξ is introduced into the optimization problem in Eq. 5.17. The variable can be introduced in two different ways:

$$\begin{cases} W_{t+1} = \underset{W \in \mathbb{R}^{K \times d}}{argmin} \frac{1}{2} \parallel W - W_t \parallel^2 + C\xi \\ \text{s.t. } l_t(W;(x_t, \tilde{Y}_t, \beta_t)) \leqslant \xi \ and \ \xi \geqslant 0 \\ W_{t+1} = \underset{W \in \mathbb{R}^{K \times d}}{argmin} \frac{1}{2} \parallel W - W_t \parallel^2 + C\xi^2 \\ \text{s.t. } l_t(W;(x_t, \tilde{Y}_t, \beta_t)) \leqslant \xi \end{cases} \tag{5.20}$$

Algorithm 5.5 (BPAs: online Bandit Passive-Aggressive algorithm for Multi-labels)**.**

Parameter: $\epsilon \in (0,1)$
Set W_1 *to the zero* $K \times d$ *matrix*
for *each* $t = 1,2,...,T$ ***do***
  Receive $\mathbf{x}_t \in \mathbb{R}^d$*;*
  for *each* $k = 1,2,\ldots,K$ ***do***
    Set $\hat{y}_t^k = \frac{1}{2}(1 + sign\langle W_t^k, x_t \rangle)$
  end for
  for *each* $k = 1,2,\ldots,K$ ***do***
    $\mathbb{P}(\tilde{y}_t^k = 1 | \hat{y}_t^k) = (1-\epsilon) \cdot \mathbb{1}_{(\hat{y}_t^k = 1)} + \epsilon \cdot \frac{\sum_{k=1}^{K} \mathbb{1}_{(\hat{y}_t^k = 1)}}{K}$
  end for
  Draw $\tilde{y}_t$ *randomly from* $P(\tilde{y}_t)$
  Receive the feedback β_t
  $l_t = \sum_{k=1}^{K} [\mathbb{1}_{\hat{y}_t^k = 1} + (1 - 2\beta_t^k)\langle W_t, \Phi(x_t, k)\rangle]_+$
  Update: $W_{t+1} = W_t + \sum_{k=1}^{K}(2\beta_t^k - 1) \cdot \frac{l_t}{\sum_{j=1}^{K}(1-2\beta_t^j)^2 \|x_t\|^2} \cdot \Phi(x_t, k)$
end for

In this part, we prove the cumulative squared loss has an upper bound. Simplify, l_t denoted $l\left(W_t;(x_t, \tilde{Y}_t, \beta_t)\right)$ and $l\left(u;(x_t, \tilde{Y}_t, \beta_t)\right)$ by l_t^*.

Theorem 5.3. *Let* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ *be a sequence of separable examples where* $x_t \in \mathbb{R}^d$*,* $Y_t \in \{0,1\}^K$ *and* $\parallel \mathbf{x}_t \parallel \leqslant R$ *for all t, and* $u \in \mathbb{R}^{K \times d}$*. Then, the cumulative squared loss of this algorithm is bounded by,*

$$\sum_{t=1}^{T} l_t^2 \leqslant K \cdot R^2 \cdot \parallel u \parallel^2 \tag{5.21}$$

Proof. Define $\Delta_t = \| W_t - u \|^2 - \| W_{t+1} - u \|^2$. $\sum_t \Delta_t$ is a telescopic sum which collapses to

$$\begin{aligned}\sum_{t=1}^{T} \Delta_t &= \sum_{t=1}^{T} \left(\| W_t - u \|^2 - \| W_{t+1} - u \|^2 \right) \\ &= \| W_1 - u \|^2 - \| W_{t+1} - u \|^2\end{aligned}$$

Considering $W_1 = \mathbf{0}$,

$$\sum_{t=1}^{T} \Delta_t = \| u \|^2 - \| W_{t+1} - u \|^2 \leqslant \| u \|^2$$

Now considering Eqs.(5.18) and (5.19),

$$\Delta_t = -2 \left\langle (W_t - u), \tau_t \sum_{k=1}^{K} (2\beta_t^k - 1)\Phi(x_t, k) \right\rangle - \left\langle \tau_t \sum_{k=1}^{K} (2\beta_t^k - 1)\Phi(x_t, k), \tau_t \sum_{k=1}^{K} (2\beta_t^k - 1)\Phi(x_t, k) \right\rangle$$

where $\tau_t = \frac{l_t}{\sum_{k=1}^{K} (2\beta_t^k - 1)^2 \| x_t \|^2}$.

Taking $l_t = \sum_{k=1}^{K} [1 + (1 - 2\beta_t^k) \langle W_t, \Phi(x_t, k) \rangle]_+$ and $l_t^* = \sum_{k=1}^{K} [1 + (1 - 2\beta_t^k) \langle u, \Phi(x_t, k) \rangle]_+$

We find

$$\Delta_t = \frac{l_t^2 - l_t l_t^*}{\sum_{k=1}^{K} (1 - 2\beta_t^k)^2 \| x_t \|^2}$$

If the examples are separable, $\exists u$ such that $\forall t \in [1, \ldots, T]$, $l_t^* = 0$,

$$\sum_{t=1}^{T} \left(\frac{l_t^2}{\sum_{k=1}^{K} (1 - 2\beta_t^k)^2 \| x_t \|^2} \right) \leqslant \| u \|^2$$

$$\sum_{t=1}^{T} l_t^2 \leqslant \sum_{k=1}^{K} (1 - 2\beta_t^k)^2 R^2 \| u \|^2$$

Because $\beta_t^k \in \{0, \frac{1}{2}, 1\}$

$$\sum_{t=1}^{T} l_t^2 \leqslant K R^2 \| u \|^2$$

■

Theorem 5.4. *Let* $(x_1, Y_1), \ldots, (x_T, Y_T)$ *be a sequence of non-separable examples where* $x_t \in \mathbb{R}^d$, $Y_t \in \{0, 1\}^K$ *and* $\| x_t \| \leqslant R$ *for all t. Then for any matrix* $u \in \mathbb{R}^{K \times d}$.

$$\sum_{t=1}^{T} l_t^2 \leqslant \left(\sqrt{K} R \| u \| + 2 \sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2 \tag{5.22}$$

Proof. By the proof of Theorem 5.4,

$$\sum_{t=1}^{T} l_t^2 \leqslant K R^2 \| u \|^2 + 2 \sum_{t=1}^{T} l_t l_t^*$$

To upper bound the right side of the above inequality, and denotes $L_t = \sqrt{\sum_{t=1}^{T} l_t^2}$ and $U_t = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}$,

$$\begin{aligned} & 2(L_t U_t)^2 - 2(\sum_{t=1}^{T} l_t l_t^*)^2 \\ = & \sum_{i=1}^{T}\sum_{j=1}^{T} l_i^2 (l_j^*)^2 + \sum_{i=1}^{T}\sum_{j=1}^{T} l_j^2 (l_i^*)^2 \\ & -2\sum_{i=1}^{T}\sum_{j=1}^{T} l_i l_j l_i^* l_j^* \\ = & \sum_{i=1}^{T}\sum_{j=1}^{T} (l_i l_j^* - l_j l_i^*)^2 \geqslant 0 \end{aligned}$$

$$\begin{aligned} \sum_{t=1}^{T} l_t^2 &\leqslant KR^2 \parallel u \parallel^2 + 2\sum_{t=1}^{T} l_t l_t^* \\ &\leqslant KR^2 \parallel u \parallel^2 + 2L_t U_t \\ (L_t - U_t)^2 &\leqslant KR^2 \parallel u \parallel^2 + U_t^2 \\ L_t &\leqslant U_t + \sqrt{KR^2 \parallel u \parallel^2 + U_t^2} \end{aligned}$$

Using the fact that $\sqrt{a+b} \leqslant \sqrt{a} + \sqrt{b}$,

$$\begin{aligned} L_t &\leqslant \sqrt{K} R \parallel u \parallel + 2U_t \\ \sum_{t=1}^{T} l_t^2 &\leqslant \left(\sqrt{K} R \parallel u \parallel + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2 \end{aligned}$$

■

5.4.1 Experiment

Data In this part, we take some experiments to evaluate some algorithms over two datasets.

Reuters RCV1-v2 collection [**?**], it is a text data set. Its label is organized by mapping the data set to the Reuters topic hierarchy, has 101 labels, and its cardinality number is 2.88. It contains 23149 instances. It is a truly high dimension data sets, the number of features D is 47236.

Yeast prepocessed by Elisseeff[**?**], where only the known structure of the functional classes are used. This multilabel dataset contains 2417 genes each represented by a 103-dimensional feature vector. There are 14 possible class labels and average cardinality is between 4.3 and 5.9.

Algorithm All participants to the experiment have been introduced, and multiclass/multilabel classification algorithm Passive-Aggressive online(PA), BPA for multilabel, and Second Order algorithm with UCB.

Evaluation Metric In multilabel classification, the predictions for an instance is a set of labels. The prediction can be fully correct, partially correct or fully wrong. That makes evaluate a multilabel classifier more challenge than multiclass classification. In this experimentation, we use the following metric to evaluate the algorithms' performance.

- **Precision** is the proportion of predicted correct labels to the total number of actual predicted labels, averaged over all instances.

$$P = \frac{1}{N}\sum_{t=1}^{N}\frac{|\,Y_t \cap \hat{Y}_t\,|}{|\,\hat{Y}_t\,|}$$

- **Recall** is the proportion of predicted correct labels to the total number of true labels, averaged over all instances.

$$R = \frac{1}{N}\sum_{t=1}^{N}\frac{|\,Y_t \cap \hat{Y}_t\,|}{|\,Y_t\,|}$$

- **One Error** determines whether the top-ranked label is the true labels, and ignores the relevancy of all other labels, averaged over all instances.

$$O = \frac{1}{N}\sum_{t=1}^{N}\mathbb{1}[\hat{y}_t^{\rho_t} \notin Y_t], \text{ where } \rho_t = \underset{i\in[K]}{argmax}\left\langle W_t^i, x_t\right\rangle$$

- **Hamming loss** reports how many times on average, the relevance of an example to a class label is incorrectly predicted. It takes into account the prediction error and the missing error.

$$H = \frac{1}{KN}\sum_{t=1}^{N}\sum_{k=1}^{K}\mathbb{1}[y_t^k \neq \hat{y}_t^k]$$

Results. Our results are summarized in Table 5.2 for the dataset RCV1-v2 and in Table 5.3 for the dataset Yeast.

Table 5.2: The summary of RCV1-v2, here Algo present Algorithm, P is precision, R is Recall, O denotes OneError, Hloss is Hamming loss and Card means Cardinality

Algo	P	R	O	Hloss	Card	Time
PA	0.629	0.185	0.07	0.021	2.001	$4.21 * 10^{-4}$
BPAs	0.990	0.159	0.151	0.027	1.37	5.37 * 10−4
2OD	0.983	0.172	0.202	0.040	1.0044	$2.13 * 10^{-1}$

5.4.2 Conclusion

Here goes the conclusion.

5.5 Conclusion

Table 5.3: The summary of Yeast dataset.

Algo	**P**	**R**	**O**	**Hloss**	**Card**	**Time**
PA	0.692	0.452	0.415	0.36	6.1	$6.12 * 10^{-7}$
BPAs	0.718	0.459	0.243	0.31	6.8	$6.5 * 10^{-7}$
2OD	0.840	0.488	0.312	0.29	5.4	$3.88 * 10^{-5}$

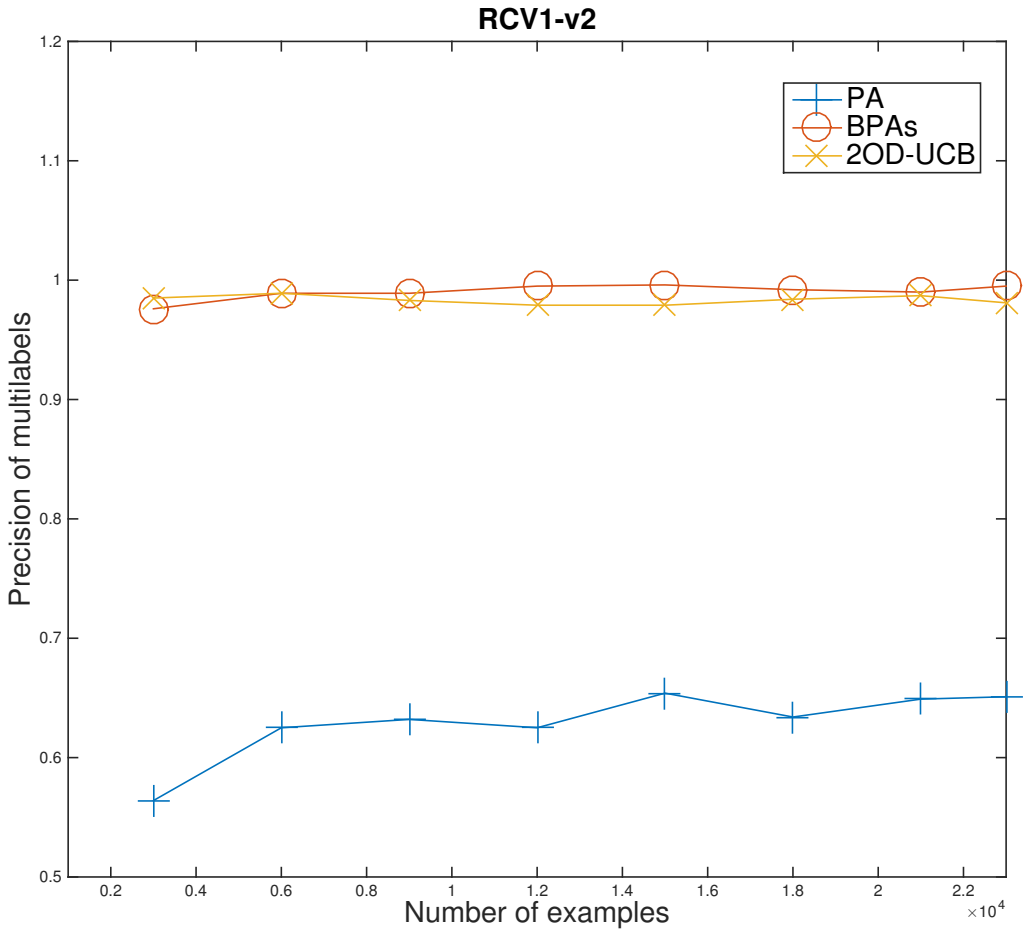

Figure 5.20: Precision of algorithms on RCV1-v2

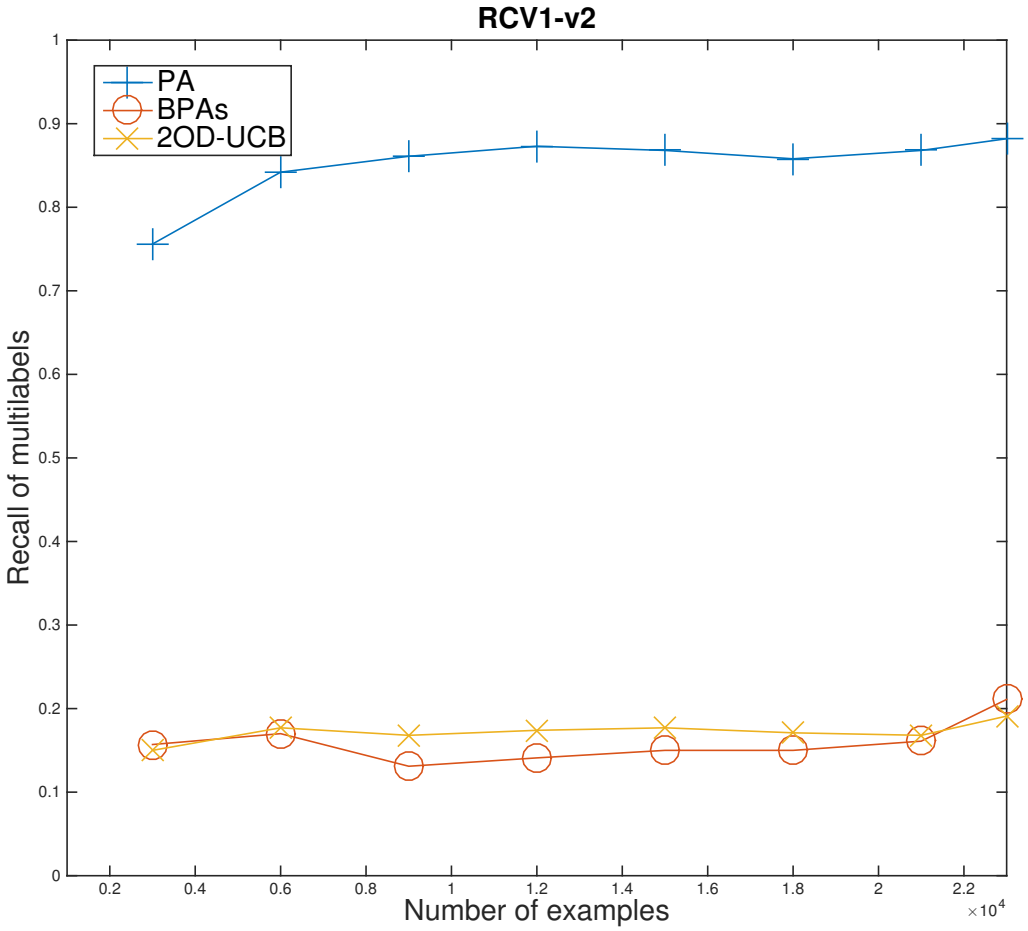

Figure 5.21: Recall of algorithms on RCV1-v2

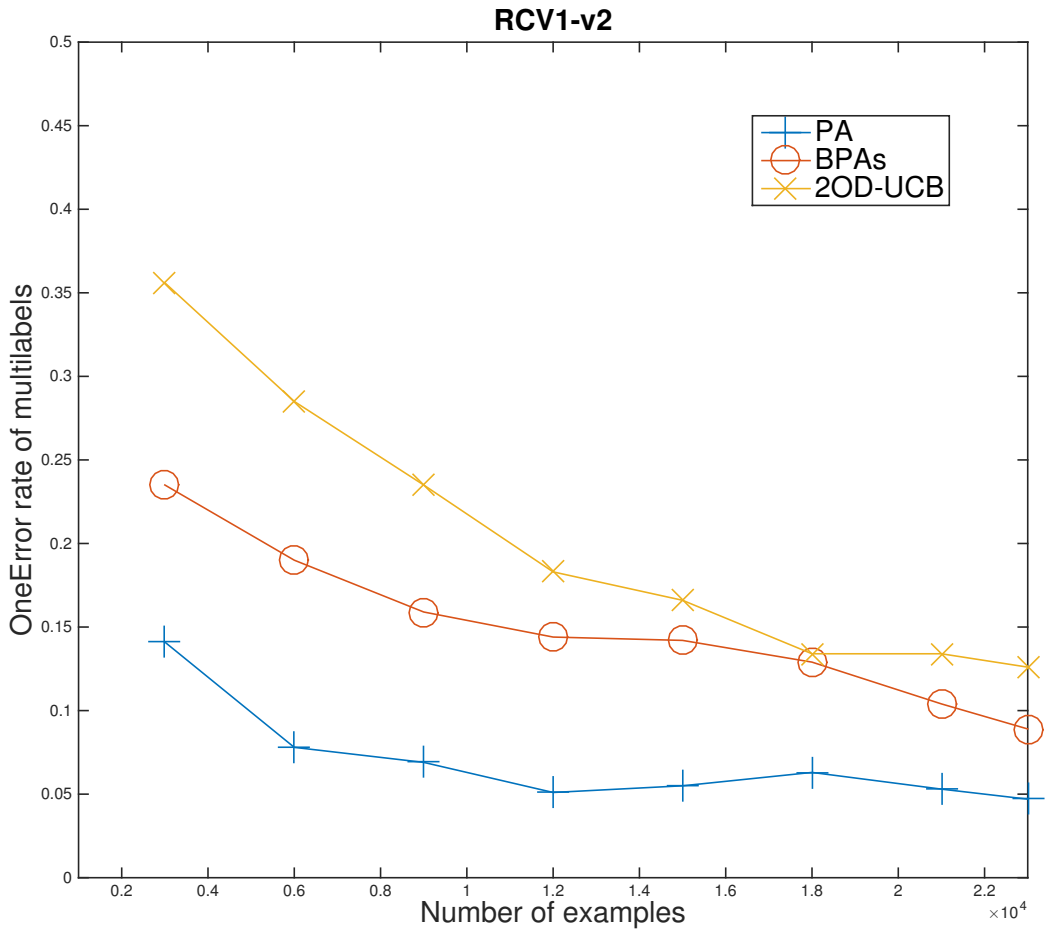

Figure 5.22: OneError of algorithms on RCV1-v2

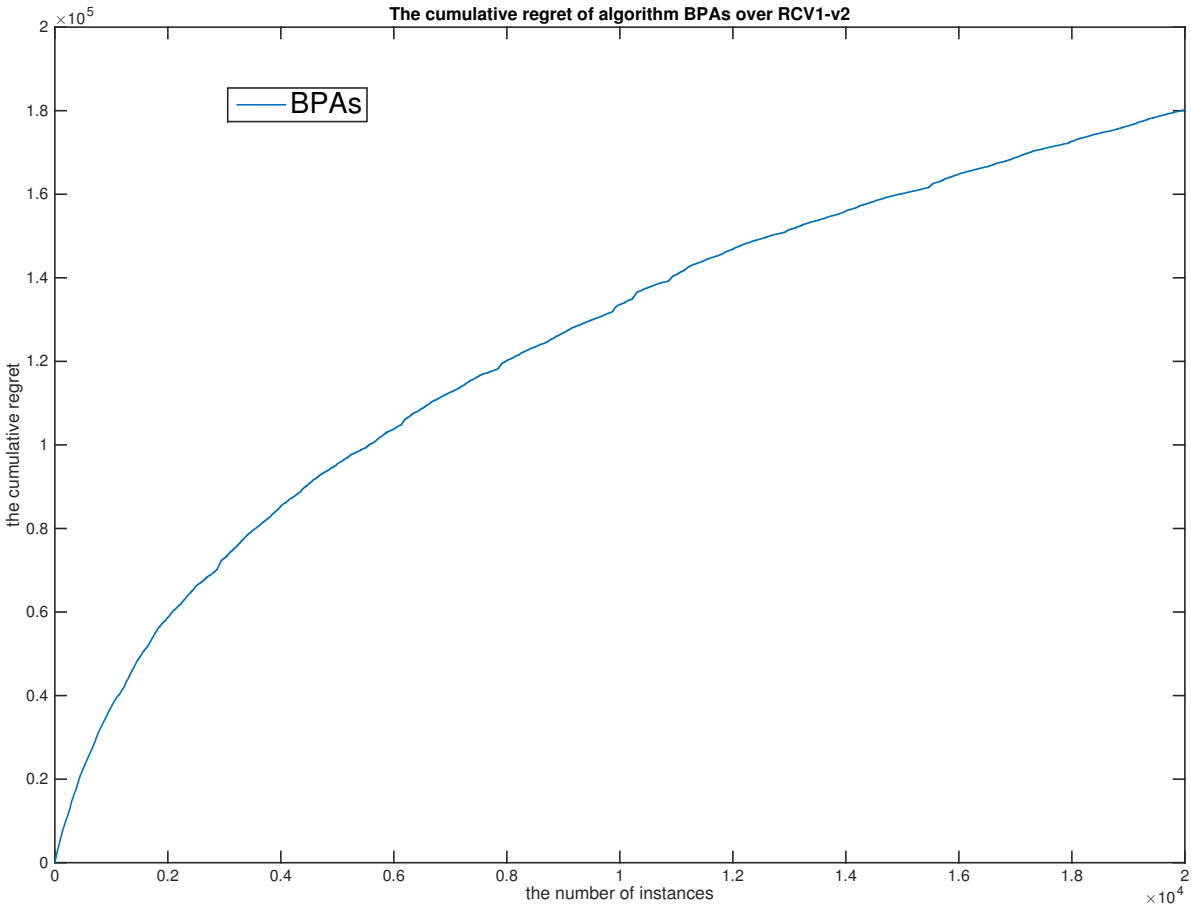

Figure 5.23: The cumulative regret of algorithms on RCV1-v2

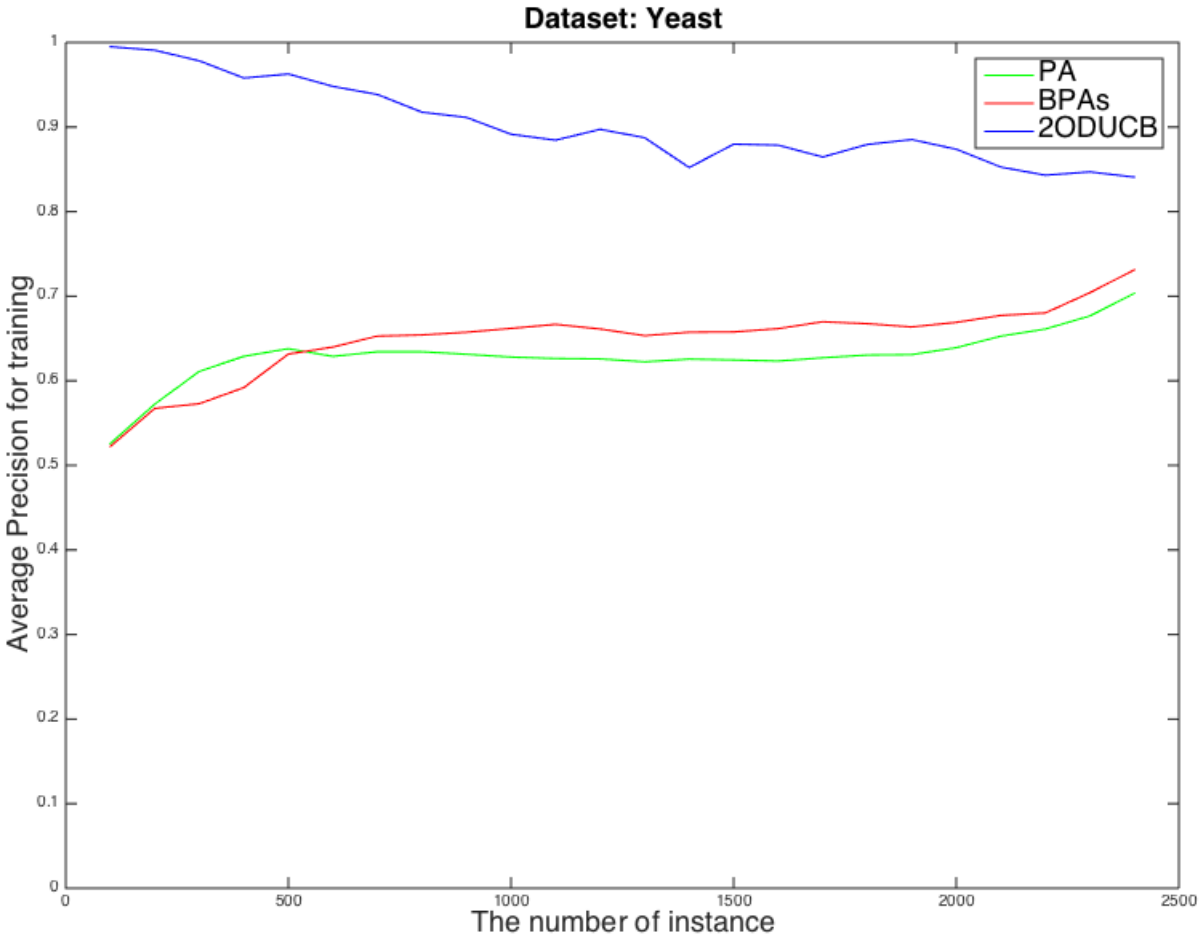

Figure 5.24: Precision of algorithms on Yeast

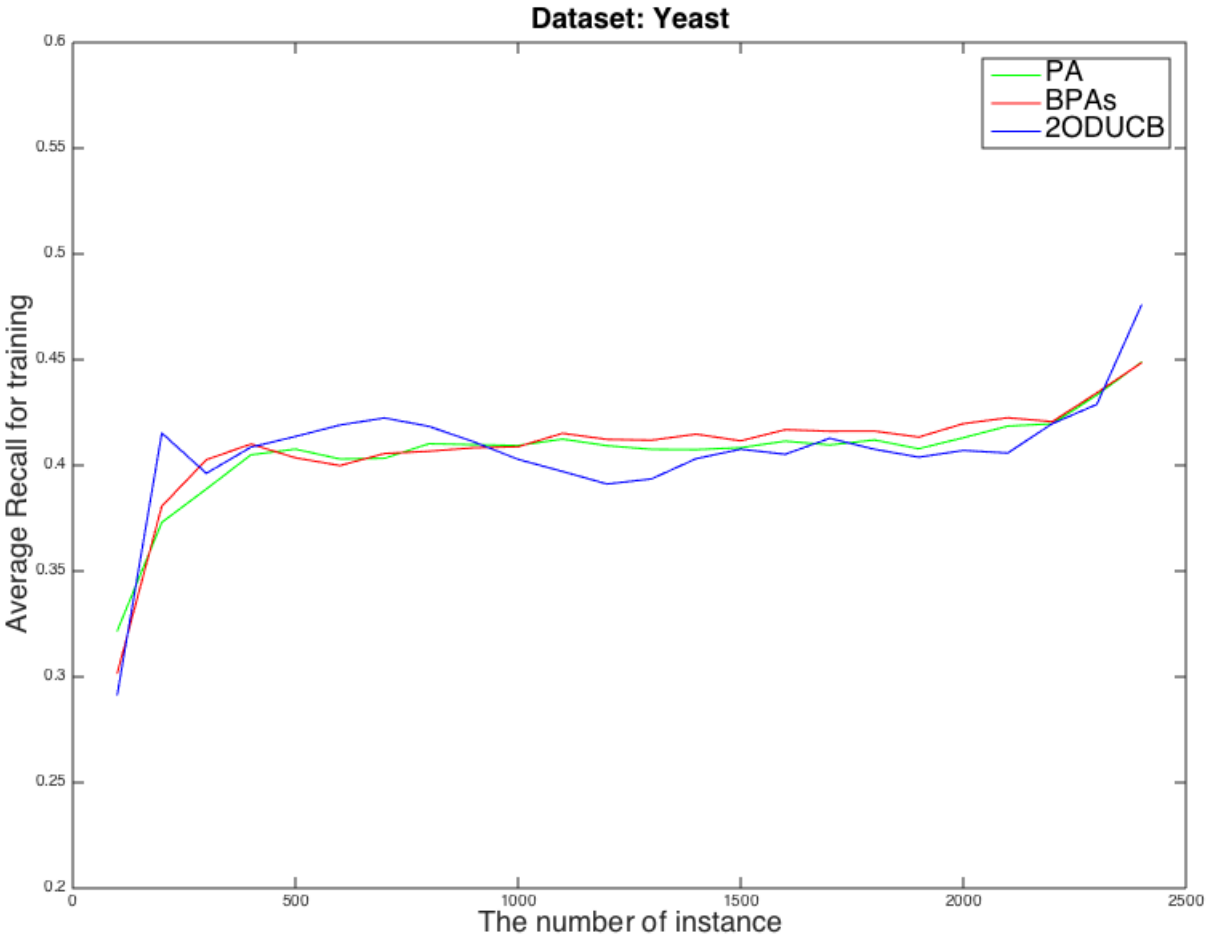

Figure 5.25: Recall of algorithms on Yeast

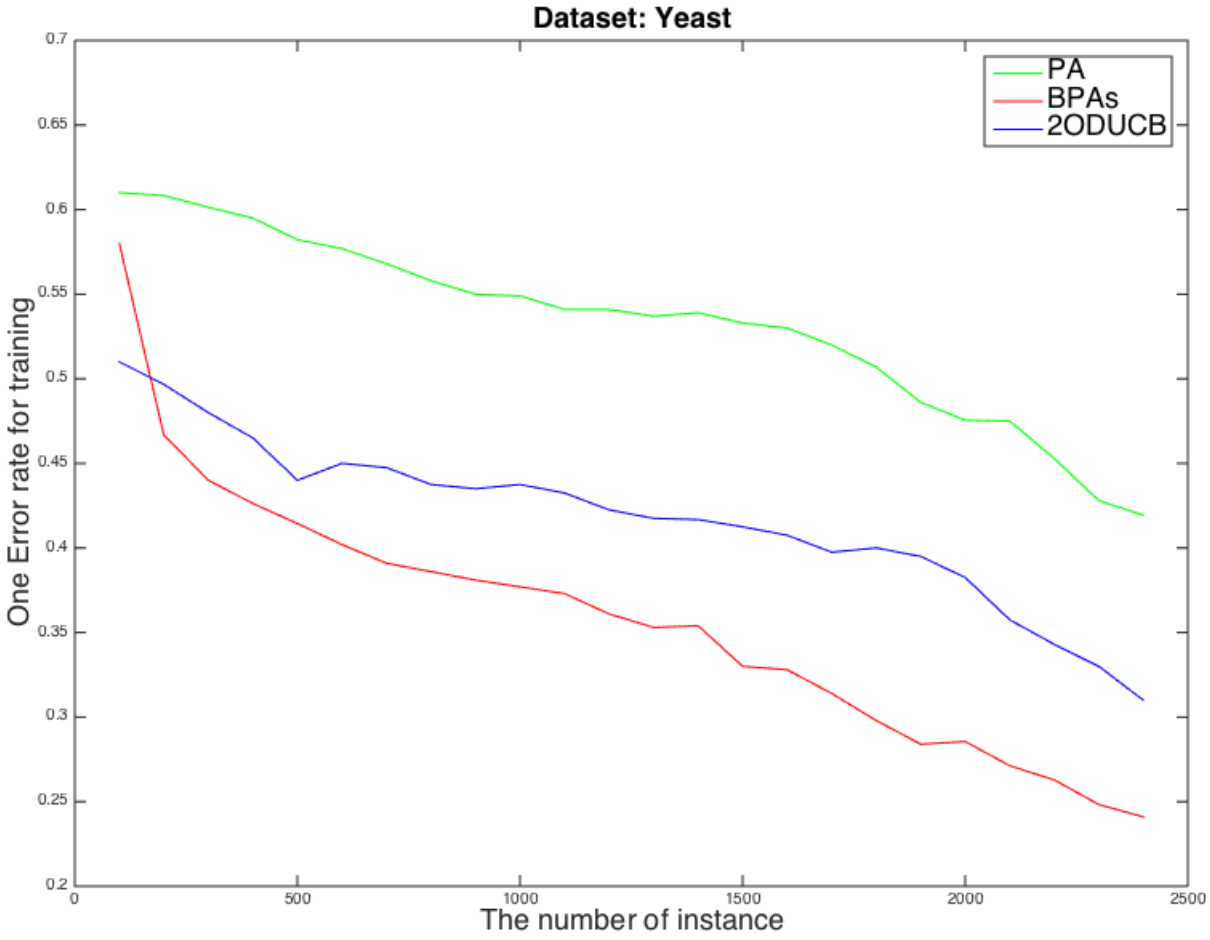

Figure 5.26: OneError of algorithms on Yeast

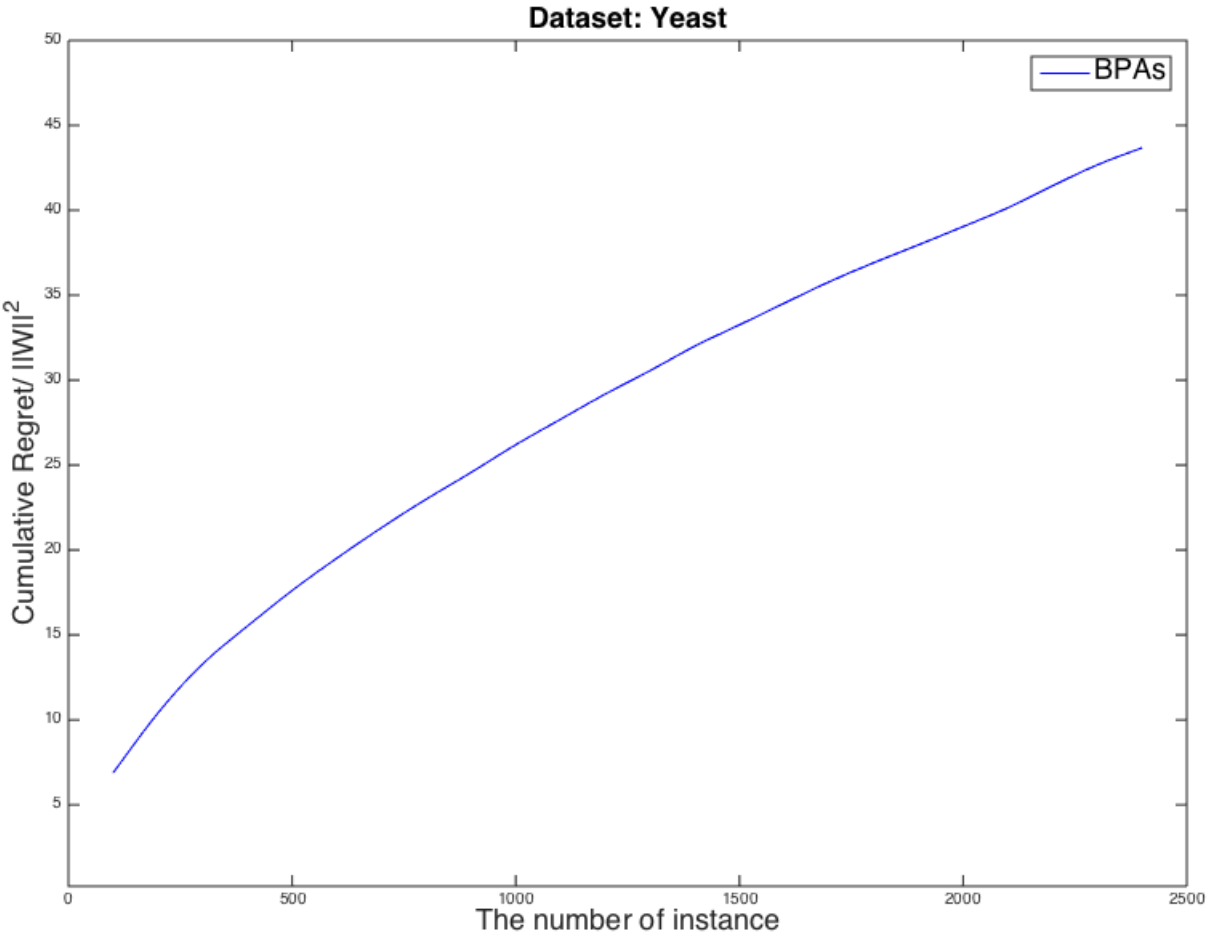

Figure 5.27: The cumulative regret / $\| W_t \|^2$ of algorithms on Yeast

CHAPTER 6

ALGORITHM OWA AND PURE FINDING WAY

Todo.

6.1 OWA way

6.2 More accuracy to identify the Pareto Front

It should combine the pure exploration way to run.

In this section, we propose an algorithm framework to find the Pareto front of Multi-Objective Multi-Armed Bandit. Referring to the finding maxima vector way in [**?**] which is least complexity, the algorithm in this paper is more effective and more accurate to find the dominated relationship in stochastic environment for MOMAB problem.

Consider an set of arms $\mathscr{A}$ with K arms, where $K \geqslant 2$, each arm has D independent objectives. At each time t, the player selects one arm i and receives a vector reward $\mathbf{r}_i$, $\mathbf{r}_i = [r_i^1, r_i^2, \ldots, r_i^D] \in \{0,1\}^K$ from a corresponding distribution with unknown mean vector $\mathbf{p}_i$, where $\mathbf{p}_i = [p_i^1, \ldots, p_i^D]$. By drawing an arm i, the player estimates the empirical mean $\hat{p}_i^d$ for the arm i in each objective $d \in D$. Let $T_i(N)$ be the number of times the arm i has been played during the first N pulls. The empirical mean $\hat{p}_i$ is the current average reward for the arm i, be estimated to $\hat{p}_i = \sum_{t=1}^{T_i(N)} \mathbf{r}_i(t)/T_i(N)$, where $\mathbf{r}_i(t)$ is the sampled values t for arm i.

Pareto Dominance and ϵ-Dominance.

In general case, there are several dominance relations that order vectors in multi-objective optimization. Pareto dominance relationship is the natural way for these environments allowing ordering the vectors directly in the multi-objective space. Let the Pareto optimal set of arms $\mathscr{A}^*$,

or the Pareto front, be the set of arms whose vector reward are non-dominated by all the other arms. All arms in the Pareto optimal set $\mathscr{A}^*$ are assumed to be equally important.

We consider that the reward vectors are ordered using the Pareto Dominance. Given two real empirical mean rewards $\hat{p}_a$ and $\hat{p}_b$ from arm a and arm b, $\hat{p}_a$ dominate $\hat{p}_b$ (denoted $\hat{p}_a \succeq \hat{p}_b$) if for $\forall i$-objective, $\hat{p}_a^i \geqslant \hat{p}_b^i$. The dominance $\hat{p}_a \succ \hat{p}_b$ is rigorous, if $\forall i$-objective, $\hat{p}_a^i > \hat{p}_b^i$. If $\hat{p}_a \not\succ \hat{p}_b$ and $\hat{p}_a \not\prec \hat{p}_b$, it means arm a and arm b can't dominate each other, they are incomparable (denoted $a \parallel b$). The set of optimal arms contains all the non-dominated arms such that $\mathscr{A}^* = \{k \in \mathscr{A} \mid \nexists k' \in \mathscr{A}, \hat{p}_{k'} \succ \hat{p}_k\}$. The Pareto front is the set of all optimal arms, where all arms are incomparable and no other arms can dominate them.

But MOMAB is a sequential stochastic learning problem. The vector $\hat{p}_i$ is an empirical mean reward to estimate the true mean reward p_i. By the stochastic environment, between $\hat{p}_i$ and p_i there is a distance ϵ, the function $(N, \epsilon) = \delta$ could explain the relationship between the distance ϵ, N the times of arm is sampled and the Confidence interval [8]. When the confidence interval is fixed, more times we sample this arm, smaller the distance ϵ.

Confidence Intervals Hoeffding's Inequality is useful to analyse the number of required samples needed to obtain a confidence interval by solving the inequality in Theorem 6.1.

Theorem 6.1 (N-dimensional Chernoff-Hoeffding bound)**.** *(Durand) Let $X_1, \ldots, X_N$ be independent D-dimensional random variables sampled with $\mathbb{E}[X_N] = [p_N^1, \ldots, p_N^D]$, $\overline{X} = \frac{1}{N}\sum_{i=1}^N X_i$, and $\mu = \mathbb{E}[X] = \frac{1}{N}\sum_{i=1}^N [p_i^1, \ldots, p_i^D]$.*

We consider the following generalization of the standard Chernoff-Hoeffding bound for D-dimensional.

$$\begin{aligned} \mathbb{P}[\overline{X} \not\prec \mu + \epsilon] &\leqslant D e^{-2N\epsilon^2} \\ \mathbb{P}[\overline{X} \not\succ \mu - \epsilon] &\leqslant D e^{-2N\epsilon^2} \end{aligned} \tag{6.1}$$

Here, we use the concept of ϵ-dominance (Trautmann, 2010) to find the Pareto front $\mathscr{A}^*$. Given two arms a and b, if for $\forall i$-objective $p_a^i - p_b^i \geqslant 2\epsilon$, and $\exists$ an objective j, $p_a^j - p_b^j > 2\epsilon$, it's said that arm a ϵ-dominates b (denoted $p_a \succ_\epsilon p_b$). If arm a non ϵ-dominate arm b and arm b non ϵ-dominate arm a, it's said that arm a is ϵ-comparable arm b (denoted $p_a \parallel_\epsilon p_b$).

In this section, we introduce the algorithm framework for finding the Pareto front in MOMAB problem. This algorithm has referred Kung's method (1975), and adapt it into the stochastic environment.

Let $\hat{p}_k(t)$ denote the mean of the observed rewards $\{\mathbf{r}_k(\tau) \mid \tau = 1, \ldots, t-1\}$.With the confidence interval, if we have an acceptable (ϵ, δ) confidence parameters, each arm we should sample $N(\epsilon, \delta)$ times. Here,

$$N = \frac{\ln(2D/\delta)}{2\epsilon^2}$$

.

After samples, we get a set of D-Objectives arms $\mathscr{A}$, it contains K-vectors, where $\mathbf{P} = \{p_1, p_2, \ldots, p_K\}$. Given a D-dimensional vector p, p^* is denoted its projection on the dimension

$[2,3,\ldots,D]$. We assume that a test for the conditions $p^* \prec_\epsilon T$ is available, where p^* is a $D-1$-dimensional vector, and T is a set of $D-1$-dimensional vectors, if $p^* \prec_\epsilon T$, it means that there is a $q \in T$ such that $p^* \prec_\epsilon q$. The first part, we introduce the algorithm with 2,3-dimension 6.1. This one has the complexity less that $O(K \log K)$. We analyze its complexity in the next section.

Algorithm 6.1 (Finding Pareto front with 2,3-Objective)**.**
Require: *Set* $T_0 = \phi \subset \mathbb{R}^{D-1}$*;*
Arrange all vectors $p \in \mathscr{A}$*,*
such that $p^1_{\sigma(1)} > p^1_{\sigma(2)} > \cdots > p^1_{\sigma(K)}$*;*
for *each* $k = 1,\ldots,K$ ***do***
if $p^1_{k-1} - p^1_k \leqslant \epsilon$ ***then***
if $p^*_k \prec_\epsilon T_{k-1}$ ***then***
$T_k \leftarrow T_{k-1}$
else
$T_k \leftarrow T_{k-1} \cup p^*_k$
end if
else
if $p^*_k \succ_\epsilon T_{k-1}$ ***then***
$T_k \leftarrow T_{k-1} \cup p^*_k$
else
$T_k \leftarrow T_{k-1}$
end if
end if
end for

Theorem 6.2. *The arm* p_k *is a Pareto optimal arm with* (ϵ,δ)*-confidence, if and only if* $p^*_k \in T_K$*.*

Proof. Presume that $T_{k-1} = \{q_1,\ldots\}$ is the Pareto front of the set $\mathscr{A}$. By the algorithm 6.1, we know that $p_k < p_{k-1}$. Since it's a MOMAB problem, we should think about the (ϵ,δ)-confidence, if p_k is an optimal arm, there is no $q \in T_{k-1}$, that $q \succ_\epsilon p_k$. When $p^1_{k-1} - p^1_k \leqslant \epsilon$, and $p^*_k \not\prec T_{k-1}$, there is no $q \in T_{k-1}$ that $\forall i \; q^i - p^i_k > \epsilon$, so p_k is an optimal arm and it will be annexed by the T_k optimal set. When $p^1_{k-1} - p^1_k > \epsilon$, if $\forall q \in T_{k-1}, \exists i = 2 or 3, p^i_k - q^i > \epsilon$. There is no vector could ϵ-dominate it, it's an optimal arm, and be annexed in the set T_k. ■

From the number of Objectives $D > 3$, the algorithm will first solve two sub-problems and then combine the results of the sub-problems, i.e., a Set of vector V is divided into two subset R and S. To find the Pareto front of V, is equivalent to find the Pareto front of R and S, then compare two sub-optimal set to find the final Pareto front for set V. The more details is shown in the Algorithm 6.3.

Algorithm 6.2 (Sub-optimal problem)**.**

Require: *Receive two sets* $R = \mathscr{A}_{i,2j-1}$ *and* $S = \mathscr{A}_{i,2j}$

Arrange *all vectors* $p \in R$ *and* $q \in S$, *such that* $p^i_{\sigma(1)} > \ldots > p^i_{\sigma(|\mathscr{A}_{i,2j-1}|)}$ *and* $q^i_{\sigma(1)} > \ldots > q^i_{\sigma(|\mathscr{A}_{i,2j}|)}$

Divide *the set* S *into two subsets* $S_1 = \{q_{\sigma(1)} \cdots, q_{\sigma(m)}\}$ *and* $S_2 = \{q_{\sigma(m+1)} \cdots, q_{\sigma(|\mathscr{A}_{i,2j}|)}\}$ *that* $m = \underset{\rho}{argmin}\rho > \left\lfloor |\mathscr{A}_{i,2j}|/2 \right\rfloor \mid \rho \in K$ *where* $K = k \mid q_k - q_{k+1} > \epsilon$

Divide *the set* R *by* q_m *into two subsets* R_1 *and* R_2 *such that* $R_1 = \{p_{\sigma(1)}, \ldots, p_{\sigma(n)}\}$ *and* $R_2 = \{p_{\sigma(n+1)}, \ldots, p_{\sigma(\mathscr{A}_{i,2j-1})}\}$ *where* $n = \underset{\rho}{argmax}\{q_{\sigma(m)} - p_{\sigma(\rho+1)} > \epsilon\}$

Then, transfer the [R, S] problem into three new sub-optimal problem $[R_1,S_1]$,$[R_2,S_1]$ *and* $[R_1,S_2]$.

Algorithm 6.3 (Finding Pareto front with D-Objective ($D > 3$))**.** ***Require:*** *Set* $i \leftarrow 1$, $\mathscr{A}_{i,j} \leftarrow \mathscr{A}$

for $D - i > 3$ *and* $|\mathscr{A}_{i,j}| > 2$ ***do***

Set $j \leftarrow 1$

for *each* $j = 1, \ldots, 2^{i-1}$ ***do***

Arrange *the vector* $p \in \mathscr{A}_{i,j}$ *by* i*-th dimension, such that* $p^i_{\sigma(1)} > \ldots > p^i_{\sigma(|\mathscr{A}_{i,j}|)}$;

Divide *the set* $\mathscr{A}_{i,j}$ *into two subsets* $\mathscr{A}_{i+1,2j-1}$ *and* $\mathscr{A}_{i+1,2j}$ *by* m, *that* $m = \underset{\rho}{argmin}\{\rho > \left\lfloor |\mathscr{A}_{i,j}|/2 \right\rfloor \mid \rho \in K\}$, *where* $K = \{k \mid p^i_k - p^i_{k+1} > \epsilon\}$;

end for

Set $i \leftarrow i + 1$

end for

for $i \neq 1$ ***do***

Set $j = 1$

for *each* $j = 1, \ldots, 2^{i-1}$ ***do***

Compare the ϵ*-dominance with all vectors of* $\mathscr{A}_{i,j}$ *from* $D-i$*-th dimension to* D*-th dimension*

if $D - i \leqslant 3$ *or* $|\mathscr{A}_{i,j}| \leqslant 2$ ***then***

Recall *the algorithm 6.1 to get the Pareto front* $\mathscr{A}^*_{i,j}$

else

Recall *the algorithm 6.2 to get the Pareto front* $\mathscr{A}^*_{i,j}$ *by two sub-Pareto set* $\mathscr{A}^*_{i+1,2j-1}$ *and* $\mathscr{A}^*_{i+1,2j}$

end if

Set $i \leftarrow i - 1$

end for

end for

6.3 Experimentation

6.4 Conclusion

Part IV

Conclusion and Perspectives

CHAPTER 7

CONCLUSION

Begins a chapter.

APPENDIX

ALGORITHMS

Multi-Armed Bandit

The strategy of trade-off

Thompson Sampling

Algorithm A.1 (Optimistic Thompson Sampling for Bernoulli Bandits)**.**

Let $\alpha_{1,k} = 1$ *and* $\beta_{1,k} = 1$*, where* $k \in \{1,\dots,K\}$

for *each round t = 1,2,..., T* ***do***

Sample $\theta_i \sim B(\alpha_{t,i}, \beta_{t,i})$*, for* $i \in \{1,\dots,K\}$

Pull arm $k_t = argmax_{k\in\{1,\dots,K\}} max\left(\theta_i, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right)$

Let $\alpha_{(t+1,k_t)} = \alpha_{(t,k_t)} + \mathbb{1}(r_{k_t}(t) = 1)$ *and* $\beta_{(t+1,k_t)} = \beta_{(t,k_t)} + \mathbb{1}(r_{k_t}(t) = 0)$.

end for

Boltzmann Exploration (Softmax)

Algorithm A.2 (Exp3)**.**

Parameter: real number $\tau > 0$ *and* $\gamma \in (0,1]$

Initialization: set $w_k(1) = 1$ *for* $k = 1,\dots,K$.

for *each round t = 1,2,..., T* ***do***

Let $p_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{i=1}^{K} w_i(t)} + \frac{\gamma}{K}$ *for* $k = 1,\dots,K$.

Pull arm k_t *s.t.* $P(k_t = k) = p_k(t)$ *for* $k = 1,\dots,K$

Receive reward $r_{k_t} \in [0,1]$

Let $\hat{r}_k(t) = r_k(t)/p_k(t)$ *if* $k = k_t$*, 0 for others.*

Let $w_k(t+1) = w_k(t)e^{\gamma \hat{r}_k(t)/K}$.

end for

Best Armed Identification in stationary MAB

Algorithm A.3 (UCB-E algorithm)**.**

Parameter: exploration parameter $a > 0$

for $k \in \{1,\dots,K\}$ ***do***

let $B_{k,s} = \hat{X}_{k,s} + \sqrt{\frac{a}{s}}$ *for* $s \geqslant 1$ *and* $B_{i,0} = +\infty$

end for

for *each round* $t = 1,\dots,T$ ***do***

Draw $k_t \in argmax_{k\in\{1,\dots,K\}} B_{k,T_k(t-1)}$

end for

Let $k_T \in argmax_{k\in\{1,\dots,K\}} \hat{X}_{k,T_k(T)}$

Algorithm A.4 (Successive Rejects algorithm)**.**

Input: $\mathscr{K}_1 = \{1,\dots,K\}$, $\overline{\log}(K) = \frac{1}{2} + \sum_{k=2}^{K} \frac{1}{k}$, $n_0 = 0$ *and for* $k \in \{1,\dots,K-1\}$

$n_k = \left\lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{K+1-k} \right\rceil$

for *for each step* $k = 1,\dots,K-1$ ***do***

for *each* $k \in \mathscr{K}_k$*, select arm* k *for* $n_k - n_{k-1}$ *rounds* ***do***

Let $\mathscr{K}_{k+1} = \mathscr{K}_k arg\ min_{k\in\mathscr{K}_k} \hat{X}_{k,n_k}$.

end for

end for

Output: the unique element of k_T *of* $\mathscr{K}$

Bandit feedback

Multiclass Classification

Algorithm A.5 (Perceptron)**.**

Initialize: Set w_1 *to the zero* $K \times d$ *matrix.*

for *each round t = 1,2,. . . ,* T ***do***

Observe $x_t \in \mathbb{R}^d$.

Predict $\hat{y}_t = \underset{\{i=1,\dots,K\}}{argmax} \langle w_t^i, x_t \rangle$

Update $w_{t+1} = w_t + (\Phi(y_t, x_t) - \Phi(\hat{y}_t, x_t))$.

end for

Algorithm A.6 (the Second-Order Perceptron)**.**

Parameter: $a > 0$

Initialize: Set $X_0 = \emptyset$*;* W_1 *to the zero* $K \times d$ *matrix;*

for *each round t = 1,2,. . . ,* T ***do***

Observe $x_t \in \mathbb{R}^n$.

Set $S_t = [X_{t-1} x_t]$

Predict $\hat{y}_t = arg\ max_{i \in \{1,\dots,K\}} < w_{i,t}, x_t >$, *where* $w_t = (aI_n + S_t S_t^T)^{-1} w_{t-1}$

Receive feedback $y_t \in \{1,\dots,K\}$

if $\hat{y}_t \neq y_t$ ***then***

$w_t = w_{t-1} + \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)$

end if

end for

Algorithm A.7 (PA algorithm in multiclass classification online learning)**.**

Initialize: Set W_1 *to the zero* $K \times d$ *matrix.*

for *each round* $t = 1,2,\dots,$ T ***do***

Observe $x_t \in \mathbb{R}^d$.

Predict $\hat{y}_t = \underset{i=\{1,\dots,K\}}{argmax} \langle W_t, \Phi(x_t, y_t) \rangle$

suffer loss: $l_t = [w_t \cdot \Phi(x_t, \hat{y}_t) - w_t \cdot \Phi(x_t, y_t) + 1]_+$

set: $\tau_t = \frac{l_t}{\|\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t)\|^2}$

Update $W_{t+1} = W_t + \tau_t (\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t))$.

end for

Banditron

Algorithm A.8 (Banditron)**.**

Parameter: number $\gamma \in \left(0, \frac{1}{2}\right)$.

Initialize: Set W_1 *to the zero* $K \times d$ *matrix.*

for *each round* $t = 1,2,\dots,$ n ***do***

Observe $x_t \in \mathbb{R}^d$.

Set $\hat{y}_t = \underset{i=1,\dots,K}{argmax} \langle W_t^i, x_t \rangle$

Prediction $Y_t \in \{1,\dots,\}$ *drawn from distribution* $p_t = \left(p_{1,t},\dots,p_{K,t}\right)$ *such that* $p_{i,t} = (1 - \gamma)\mathbb{1}_{\hat{y}_t = i} + \frac{\gamma}{K}$.

Observe $\mathbb{1}_{(\hat{y}_t = y_t)}$.

Update $W_{t+1} = W_t + \left(\frac{\mathbb{1}_{\tilde{y}_t = y_t}}{p_{i,t}} \Phi(\tilde{y}_t, x_t) - \Phi(\hat{y}_t, x_t)\right)$.

end for

Algorithm A.9 (Confidit)**.**

Parameter: $\alpha \in (-1, 1]$

Initialization: $A_0 = (1 + \alpha)^2 I \in \mathbb{R}^{dK \times dK}$, $W_o = (\mathbf{w}_{1,0},\dots,\mathbf{w}_{K,0}) = \mathbf{0} \in \mathbb{R}^{dK}$;

for *each round* $(\mathbf{x}_1, y_1),\dots,(\mathbf{x}_T, y_T)$ ***do***

Get instance $\mathbf{x}_t \in \mathbb{R}^d$, *and normalized it* $\| \mathbf{x}_t \| = 1$;

Set:

$$W'_{t-1} = \underset{W \in \mathbb{R}^{dK}}{argmin} d_{t-1}(W, W_{t-1})$$
$$s.t. \forall i \in [K] - \alpha \leqslant \mathbf{w}_i^T \mathbf{x}_t \; and \; \sum_{i=1}^{K} \mathbf{w}_i^T \mathbf{x}_t = 1 + \alpha - K\alpha$$

Set: $\forall i \in [K] \hat{\Delta}'_{i,t} = \mathbf{x}_t^T \mathbf{w}'_{i,t-1}$;
Output:

$$\hat{y}_t = arg\ max_i(\hat{\Delta}'_{i,t} + \epsilon_{i,t}), where \epsilon_{i,t}^2 = \left(2\mathbf{x}_t^T A_{i,t}^{-1} \mathbf{x}_t\right) \times \eta_t$$
$$and \eta_t = \frac{1}{2}(1+\alpha)^2 \parallel U \parallel_2^2 + \frac{(1+\alpha)^2}{2} \sum_{s=1}^{t-1} \mathbf{x}_s^T A_{\hat{y}_s,s}^{-1} \mathbf{x}_s + 9(1+\alpha)^2 \log \frac{t+4}{\delta}$$

Get feedback $M_t = \{y_t \neq \hat{y}_t\}$;
if $M_t = 1$ ***then***
with probability $(1-\alpha)/2$ *set*

$$X_t = (0, \ldots, 0, \underbrace{\mathbf{x}_t}_{position\ \hat{y}_t}, 0, \ldots, 0)$$

with probability $(1+\alpha)/2$ *set*

$$X_t = (0, \ldots, 0, \underbrace{-\mathbf{x}_t}_{position\ \hat{y}_t}, 0, \ldots, 0)$$

else

$$X_t = (0, \ldots, 0, \underbrace{\mathbf{x}_t}_{position\ \hat{y}_t}, 0, \ldots, 0)$$

end if
Update:

$$A_t = A_{t-1} + X_t X_t^T$$
$$W_t = A_t^{-1}(A_{t-1} W'_{t-1} + X_t).$$

end for

Multi-labels Classification

Multi-labels Classification in bandit setting

Algorithm A.10 (The algorithm based on 2nd order in bandit setting)**.**

Parameters: loss parameter $a \in [0,1]$*, cost value* $c(i,s)$*, interval* $D = [-R,R]$*, function* $g \to R$*, confidence level* $\delta \in [0,1]$

Initialization: $A_{i,0} = I \in \mathbb{R}^{d \times d}$*,* $i = 1,\ldots,K$*,* $w_{i,1} = \in \mathbb{R}^d$*,* $i = 1,\ldots,K$*;*

for *for each instance* $(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_T, y_T)$ ***do***

Get instance $\mathbf{x}_t \in \mathbb{R}^d$*:* $\| \mathbf{x}_t \|^2 = 1$*;*

$\forall i \in [K]$*, set* $\hat{\Delta}'_{i,t} = \mathbf{x}_t^T w'_{i,t}$*, where*

$$w'_{i,t} = \begin{cases} w_{i,t} & \text{if } w_{i,t}^T \mathbf{x}_t \in [-R,R], \\ w_{i,t} - \left(\frac{w_{i,t}^T \mathbf{x}_t - R sign(w_{i,t}^T \mathbf{x}_t)}{\mathbf{x}_t^T A_{i,t-1}^{-1} \mathbf{x}_t} \right) A_{i,t-1}^{-1} \mathbf{x}_t & \text{otherwise}; \end{cases}$$

Output

$$\hat{Y}_t = arg\ min_{Y=(j_1,j_2,\ldots,j_{|Y|}) \subseteq [K]} \left(\sum_{i \in Y} (c(j_i,|Y|) - (\frac{a}{1-a} + c(j_i,|Y|)) \hat{p}_{i,t}) \right)$$

, where: $\hat{p}_{i,t} = \frac{g(-[\hat{\Delta}'_{i,t} + \epsilon_{i,t}]_D)}{g(-[\hat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) + g([\hat{\Delta}'_{i,t} + \epsilon_{i,t}]_D)}$*, and* $\epsilon_{i,t}^2 = \eta \mathbf{x}_t^T A_{i,t}^{-1} \mathbf{x}_t$

Get bandit feedback $Y_t \cap \hat{Y}_t$*;*

$\forall i \in [K]$*, update* $A_{i,t} = A_{i,t-1} + |s_{i,t}| \mathbf{x}_t \mathbf{x}_t^T$*,* $w_{i,t+1} = w'_{i,t} - \frac{1}{c''_L} A_{i,t}^{-1} \bigtriangledown_{i,t}$*, where*

$$s_{i,t} = \begin{cases} 1 & \text{if } i \in Y_t \cap \hat{Y}_t \\ -1 & \text{if } i \in \hat{Y}_t \setminus Y_t = \hat{Y}_t \setminus (Y_t \cap \hat{Y}_t) \\ 0 & \text{otherwise}; \end{cases}$$

with $\bigtriangledown_{i,t} = -g(s_{i,t} \hat{\Delta}_{i,t'}) s_{i,t} \mathbf{x}_t$*.*

end for

Multi-Objective Multi-Armed Bandit

Dominance method

Algorithm A.11 (Global SEMO)**.**

Choose $x \in \mathbb{B}^n$ *uniformly at random.*

Initialize $P := \{x\}$*.*

repeat

Choose $x \in P$ *uniformly at random*

Create an offspring y *by flipping each bit of* x *with probability* $1/n$*.*

if $\{z \in P | z \succ y\} = \emptyset$ ***then***

Update $P := (P \{z \in P | y \succeq z\}) \cup \{y\}$

end if

until *all* $x \in \mathbb{B}^n$ *be chosen.*

Algorithm A.12 (Global DEMO$_\epsilon$)**.**

Choose $x \in \mathbb{B}^n$ uniformly at random.
Initialize $P := \{x\}$.
repeat
Choose $x \in P$ uniformly at random
Create an offspring y by flipping each bit of x with probability $1/n$.
if $\{z \in P | b(z) \succ b(y) \vee z \succ y\} = \emptyset$ ***then***
Update $P := (P\{z \in P | b(y) \succeq bz\}) \cup \{y\}$
end if
until *all $x \in \mathbb{B}^n$ be chosen.*

Multi-Objective Multi-Armed Bandit

Algorithm A.13 (The scalarized PAC algorithm sPAC(ϵ, δ, W))**.**

for *all arms $k = \{1, \ldots, K\}$* ***do***
Pull each arm k for $l = \frac{1}{(\epsilon/2)^2} \log(\frac{2|W|K}{\delta})$ times:
Compute the expected mean reward vectors $\hat{\mu}_k$
end for
Initiates $\mathscr{A}^* \rightarrow \emptyset$;
for *all weight vectors $w \in W$* ***do***
select an optimal arm i^* for function f_w;
Add arm i^* to the Pareto front $\mathscr{A}^* \rightarrow \mathscr{A}^* \cup \{i^*\}$
Delete dominated arms from $\mathscr{A}^*$
end for
return $\mathscr{A}^*$

Algorithm A.14 (Annealing Scalarized Algorithm)**.**

Input: number of arms $|\mathscr{A}|$, horizon of times T, number of objectives $|d|$, set of linear scalarized function $S = \{f^1, f^2, \ldots, f^{|S|}\}$, decay parameter $\epsilon_0 \in (0,1)$, the reward distribution.
Initialize: for each scalarized function $s = 1$ to S, each arm i played initial times to get the estimated vector $\hat{\mu}_i^s$, set annealing set $\mathscr{A}_\epsilon^s = \mathscr{A}$
for *time step $t = 1, \ldots, T$* ***do***
set the parameter $\epsilon_t = \epsilon_0^t / (|\mathscr{A}||d|)$
select f^s uniformly at random
Compute: the weight set $w^s \rightarrow (w^{1,s}, \ldots, w^{d,s})$
$f^s(\hat{\mu}^s) = max_{1 \leqslant i \leqslant |\mathscr{A}|} f^s(\hat{\mu}^s)$
for *arm $i = 1, \ldots, |\mathscr{A}|$* ***do***
if $f^s(\hat{\mu}^s) \in [f^s(\hat{\mu}^s) - \epsilon_t, f^s(\hat{\mu}^s)]$ ***then***
$\mathscr{A}_\epsilon^*(t) \rightarrow \mathscr{A}_\epsilon^*(t) \cup i$
end if

end for

$S_{difference} \leftarrow (\mathscr{A}_\epsilon^*(t-1)) - \mathscr{A}_\epsilon^*(t)$

for *arm* $j \in S_{difference}$ ***do***

if $\hat{\mu}_k \not\succ \hat{\mu}_j, \forall k \in \mathscr{A}$ ***then***

$\mathscr{A}_\epsilon^*(t) \leftarrow \mathscr{A}_\epsilon^*(t) \cup j$

end if

end for

$(\mathscr{A}^*)^s(t-1) \leftarrow \mathscr{A}_\epsilon^*(t)$

Pull an optimal arm $(i^*)^s$ *from* $(A_\epsilon^*)^s$

Observe: $r_{i^*}^s$*;*

Update: $\hat{\mu}_{i^*}^s$

end for

Appendix B

Programming

BIBLIOGRAPHY

[1] Y. ABBASI-YADKORI, D. PÁL, AND C. SZEPESVÁRI, *Improved algorithms for linear stochastic bandits*, in Advances in Neural Information Processing Systems, 2011, pp. 2312–2320.

[2] R. AGRAWAL, *Sample mean based index policies with o (log n) regret for the multi-armed bandit problem*, Advances in Applied Probability, (1995), pp. 1054–1078.

[3] S. AGRAWAL AND N. GOYAL, *Analysis of thompson sampling for the multi-armed bandit problem*, arXiv preprint arXiv:1111.1797, (2011).

[4] ——, *Thompson sampling for contextual bandits with linear payoffs*, arXiv preprint arXiv:1209.3352, (2012).

[5] C. M. ANDERSON, *Behavioral models of strategies in multi-armed bandit problems*, PhD thesis, California Institute of Technology, 2001.

[6] J. AUDIBERT AND S. BUBECK, *Best arm identification in multi-armed bandits*, in COLT-23th Conference on Learning Theory-2010, 2010, pp. 13–p.

[7] P. AUER, *Using confidence bounds for exploitation-exploration trade-offs*, The Journal of Machine Learning Research, 3 (2003), pp. 397–422.

[8] P. AUER, N. CESA-BIANCHI, AND P. FISCHER, *Finite-time analysis of the multiarmed bandit problem*, Machine learning, 47 (2002), pp. 235–256.

[9] P. AUER, N. CESA-BIANCHI, Y. FREUND, AND R. SCHAPIRE, *The nonstochastic multiarmed bandit problem*, SIAM Journal on Computing, 32 (2003), pp. 48–77.

[10] P. AUER, N. CESA-BIANCHI, Y. FREUND, AND R. E. SCHAPIRE, *Gambling in a rigged casino: The adversarial multi-armed bandit problem*, in Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on, IEEE, 1995, pp. 322–331.

[11] P. AUER AND R. ORTNER, *Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem*, Periodica Mathematica Hungarica, 61 (2010), pp. 55–65.

[12] J. BANKS, M. OLSON, AND D. PORTER, *An experimental analysis of the bandit problem*, Economic Theory, 10 (1997), pp. 55–77.

[13] R. BELLMAN, *A markovian decision process*, tech. rep., DTIC Document, 1957.

[14] R. E. BELLMAN AND L. A. ZADEH, *Decision-making in a fuzzy environment*, Management science, 17 (1970), pp. B–141.

[15] D. A. BERRY AND B. FRISTEDT, *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*, Springer, 1985.

[16] K. BRINKER, J. FÜRNKRANZ, AND E. HÜLLERMEIER, *Label ranking by learning pairwise preferences*, tech. rep., Citeseer, 2007.

[17] D. BROCKHOFF, T. FRIEDRICH, N. HEBBINGHAUS, C. KLEIN, F. NEUMANN, AND E. ZITZLER, *Do additional objectives make a problem harder?*, in Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, 2007, pp. 765–772.

[18] S. BUBECK, R. MUNOS, AND G. STOLTZ, *Pure exploration in multi-armed bandits problems*, in Algorithmic Learning Theory, Springer, 2009, pp. 23–37.

[19] ——, *Pure exploration in finitely-armed and continuous-armed bandits*, Theoretical Computer Science, 412 (2011), pp. 1832–1852.

[20] A. CARPENTIER AND R. MUNOS, *Bandit theory meets compressed sensing for high dimensional stochastic linear bandit*, arXiv preprint arXiv:1205.4094, (2012).

[21] N. CESA-BIANCHI, A. CONCONI, AND C. GENTILE, *A second-order perceptron algorithm*, SIAM Journal on Computing, 34 (2005), pp. 640–668.

[22] N. CESA-BIANCHI AND P. FISCHER, *Finite-time regret bounds for the multiarmed bandit problem.*, in ICML, Citeseer, 1998, pp. 100–108.

[23] O. CHAPELLE AND L. LI, *An empirical evaluation of thompson sampling*, in Advances in neural information processing systems, 2011, pp. 2249–2257.

[24] S. CHEN, T. LIN, I. KING, M. R. LYU, AND W. CHEN, *Combinatorial pure exploration of multi-armed bandits*, in Advances in Neural Information Processing Systems, 2014, pp. 379–387.

[25] F. O. CLAUDIO GENTILE, *On multilabel classification and ranking with bandit feedback*, Journal of Machine Learning Research, (2014).

[26] K. CRAMMER, O. DEKEL, J. KESHET, S. SHALEV-SHWARTZ, AND Y. SINGER, *Online passive-aggressive algorithms*, The Journal of Machine Learning Research, 7 (2006), pp. 551–585.

[27] K. CRAMMER AND C. GENTILE, *Multiclass classification with bandit feedback using adaptive regularization*, Mach Learn, 90 (2013), pp. 347–383.

[28] K. CRAMMER AND Y. SINGER, *Ultraconservative online algorithms for multiclass problems*, The Journal of Machine Learning Research, 3 (2003), pp. 951–991.

[29] N. D. DAW, J. P. O'DOHERTY, P. DAYAN, B. SEYMOUR, AND R. J. DOLAN, *Cortical substrates for exploratory decisions in humans*, Nature, 441 (2006), pp. 876–879.

[30] K. DEB, *Multi-objective optimization using evolutionary algorithms*, vol. 16, John Wiley & Sons, 2001.

[31] M. DRUGAN AND A. NOWE, *Designing multi-objective multi-armed bandits algorithms: A study*, in Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–8.

[32] M. M. DRUGAN, *Linear scalarization for pareto front identification in stochastic environments*, in Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 156–171.

[33] N. EHSAN AND M. LIU, *On the optimality of an index policy for bandwidth allocation with delayed state observation and differentiated services*, in INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 3, IEEE, 2004, pp. 1974–1983.

[34] E. EVEN-DAR, S. MANNOR, AND Y. MANSOUR, *Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems*, The Journal of Machine Learning Research, 7 (2006), pp. 1079–1105.

[35] W. GAO AND Z.-H. ZHOU, *On the consistency of multi-label learning*, Artificial Intelligence, 199 (2013), pp. 22–44.

[36] A. GARIVIER AND E. MOULINES, *On upper-confidence bound policies for non-stationary bandit problems*, Arxiv preprint, (2008).

[37] O. GIEL AND P. K. LEHRE, *On the effect of populations in evolutionary multi-objective optimisation**, Evolutionary Computation, 18 (2010), pp. 335–356.

[38] J. GITTINS AND D. JONES, *A Dynamic Allocation Index for the Sequential Allocation of Experiments*, Progress in Statistics, 1974.

[39] J. C. GITTINS, *Bandit processes and dynamic allocation indices*, Journal of the Royal Statistical Society. Series B (Methodological), (1979), pp. 148–177.

[40] J. C. GITTINS AND D. M. JONES, *A dynamic allocation index for the discounted multiarmed bandit problem*, Biometrika, 66 (1979), pp. 561–565.

[41] C. HOROBA AND F. NEUMANN, *Benefits and drawbacks for the use of epsilon-dominance in evolutionary multi-objective optimization*, in Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM, 2008, pp. 641–648.

[42] L. P. KAELBLING, M. L. LITTMAN, AND A. W. MOORE, *Reinforcement learning: A survey*, Journal of artificial intelligence research, (1996), pp. 237–285.

[43] S. M. KAKADE, S. SHALEV-SHWARTZ, AND A. TEWARI, *Efficient bandit algorithms for online multiclass prediction*, in Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 440–447.

[44] E. KAUFMANN, N. KORDA, AND R. MUNOS, *Thompson sampling: An asymptotically optimal finite-time analysis*, in Algorithmic Learning Theory, Springer, 2012, pp. 199–213.

[45] J. KIVINEN AND M. K. WARMUTH, *Additive versus exponentiated gradient updates for linear prediction*, in Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, ACM, 1995, pp. 209–218.

[46] X. KONG AND P. S. YU, *An ensemble-based approach to fast classification of multi-label data streams*, in Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on, IEEE, 2011, pp. 95–104.

[47] T. L. LAI AND H. ROBBINS, *Asymptotically efficient adaptive allocation rules*, Advances in applied mathematics, 6 (1985), pp. 4–22.

[48] J. LANGFORD AND T. ZHANG, *The epoch-greedy algorithm for multi-armed bandits with side information*, in Advances in neural information processing systems, 2008, pp. 817–824.

[49] N. LITTLESTONE, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*, Machine learning, 2 (1988), pp. 285–318.

[50] E. LOZA MENCÍA AND J. FURNKRANZ, *Pairwise learning of multilabel classifications with perceptrons*, in Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, IEEE, 2008, pp. 2899–2906.

[51] R. D. LUCE, *Individual choice behavior: A theoretical analysis*, Courier Corporation, 1959.

[52] G. MADJAROV, D. KOCEV, D. GJORGJEVIKJ, AND S. DŽEROSKI, *An extensive experimental comparison of methods for multi-label learning*, Pattern Recognition, 45 (2012), pp. 3084–3104.

[53] B. C. MAY, N. KORDA, A. LEE, AND D. S. LESLIE, *Optimistic bayesian sampling in contextual-bandit problems*, The Journal of Machine Learning Research, 13 (2012), pp. 2069–2106.

[54] K. MIETTINEN, *Nonlinear multiobjective optimization*, vol. 12, Springer Science & Business Media, 2012.

[55] M. L. MINSKY AND S. A. PAPERT, *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*, MIT press Boston, MA:, 1987.

[56] V. PARETO, *Cours d'economies politique, volume ,Ö† and ,Ö°*, F Rouge, Lausanne, (1896).

[57] H. ROBBINS, *Some aspects of the sequential design of experiments*, Bulleting of the American Mathematical Society, 58 (1952), pp. 527–535.

[58] F. ROSENBLATT, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), p. 386.

[59] S. L. SCOTT, *A modern bayesian look at the multi-armed bandit*, Applied Stochastic Models in Business and Industry, 26 (2010), pp. 639–658.

[60] M. S. SOROWER, *A literature survey on algorithms for multi-label learning*, Oregon State University, Corvallis, (2010).

[61] M. STEYVERS, M. D. LEE, AND E.-J. WAGENMAKERS, *A bayesian analysis of human decision-making on bandit problems*, Journal of Mathematical Psychology, 53 (2009), pp. 168–179.

[62] J. SUROWIECKI, *The wisdom of crowds*, Anchor, 2005.

[63] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*, vol. 1, MIT press Cambridge, 1998.

[64] W. R. THOMPSON, *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*, Biometrika, (1933), pp. 285–294.

[65] M. TOKIC, *Adaptive ε-greedy exploration in reinforcement learning based on value differences*, in KI 2010: Advances in Artificial Intelligence, Springer, 2010, pp. 203–210.

[66] J. N. TSITSIKLIS, *A short proof of the gittins index theorem*, The Annals of Applied Probability, (1994), pp. 194–199.

[67] P. VARAIYA, J. WALRAND, AND C. BUYUKKOC, *Extension of the multi-armed bandit problem*, in Decision and Control, 1983. The 22nd IEEE Conference on, IEEE, 1983, pp. 1179–1180.

[68] M. H. VEATCH AND L. M. WEIN, *Scheduling a make-to-stock queue: Index policies and hedging points*, Operations Research, 44 (1996), pp. 634–647.

[69] R. WEBER ET AL., *On the gittins index for multiarmed bandits*, The Annals of Applied Probability, 2 (1992), pp. 1024–1033.

[70] P. WHITTLE, *Multi-armed bandits and the gittins index*, Journal of the Royal Statistical Society. Series B (Methodological), (1980), pp. 143–149.

[71] R. R. YAGER, *On ordered weighted averaging aggregation operators in multicriteria decision-making*, Systems, Man and Cybernetics, IEEE Transactions on, 18 (1988), pp. 183–190.

[72] ——, *Quantifier guided aggregation using owa operators*, International Journal of Intelligent Systems, 11 (1996), pp. 49–73.

[73] S. YAHYAA, M. DRUGAN, AND B. MANDERICK, *The scalarized multi-objective multi-armed bandit problem: an empirical study of its exploration vs. exploitation tradeoff*, in Neural Networks (IJCNN), 2014 International Joint Conference on, IEEE, 2014, pp. 2290–2297.

[74] L. A. ZADEH, *A computational theory of dispositions*, in Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, 1984, pp. 312–318.

[75] E. ZITZLER, L. THIELE, M. LAUMANNS, C. M. FONSECA, AND V. G. DA FONSECA, *Performance assessment of multiobjective optimizers: an analysis and review*, Evolutionary Computation, IEEE Transactions on, 7 (2003), pp. 117–132.