Bandit feedback in Classification and Multi-objective Optimization

Hongliang Zhong

CENTRALE MARSEILLE

École Doctorale : Mathématique et Informatique de Marseille (ED184)

Laboratoire d'Informatique Fondamentale

THÈSE DE DOCTORAT

pour obtenir le grade de

DOCTEUR de l'ÉCOLE CENTRALE de MARSEILLE

Discipline : Informatique

Bandit feedback in Classification and Multi-objective Optimization

par

ZHONG Hongliang

Directeurs de thèse : RALAIVOLA Liva, DAUCÉ Emmanuel

Soutenue le 29 Mars 2016

devant le jury composé de :

PREUX Philippe	INRIA LNE Equipe Sequel	rapporteurs
GASSO Gilles	INSA ROUEN Laboratoire LITIS EA 4108	rapporteurs
DUTECH Alain	INRIA Equipe MAIA/LPRIA	examinateur
ARTIÈRES Thierry	ECM Equipe QARMA	examinateur
RALAIVOLA Liva	AMU Laboratoire d'Informatique Fondamentale	examinateur
DAUCÉ Emmanuel	ECM Institut de Neurosciences des systemes	examinateur

Abstract

Bandit problems constitute a sequential dynamic allocation problem. The pulling agent has to explore its environment (i.e. the arms) to gather information on the one hand, and it has to exploit the collected clues to increase its rewards on the other hand. How to adequately balance the exploration phase and the exploitation phase is the crux of bandit problems and most of the efforts devoted by the research community from this fields has focused on finding the right exploitation/exploration tradeoff. In this dissertation, we focus on investigating two specific bandit problems: the contextual bandit problem and the multi-objective bandit problem.

This dissertation provides two contributions. The first contribution is about the classification under partial supervision, which we encode as a contextual bandit problem with side information. This kind of problem is heavily studied by researchers working on social networks and recommendation systems. We provide a series of algorithms to solve the Bandit feedback problem that pertain to the Passive-Aggressive family of algorithms. We take advantage of its grounded foundations and we are able to show that our algorithms are much simpler to implement than state-of-the-art algorithms for bandit with partial feedback, and they yet achieve better performances of classification. For multi-objective multi-armed bandit problem (MOMAB), we propose an effective and theoretically motivated method to identify the Pareto front of arms. We in particular show that we can find all elements of the Pareto front with a minimal budget.

Key words

Bandit feedback, Classification, Passive-Aggressive algorithm, Front Pareto, Multi-objective Multi-armed Bandit

Dedication and acknowledgements

I have received so much help and support from many people for my dissertation.

Firstly, I would like to thank my supervisors, Prof. Liva Ralaivola and Prof. Emmanuel Daucé. Thanks deeply for their guidance that allowed me to complete this thesis. They not only give me the advice and show me the theory, but also teach me how to research it myself.

I also thank the reviewers of my dissertation: Prof. Philippe Preux and Prof. Gilles Gasso for their helpful comments.

Thank you to my families, especially my wife for all support.

Research in this dissertation was supported by China Scholarship Council and partly by ANR-funded projet GRETA.

Résumé en français

Des problèmes de Bandit constituent une séquence d'allocation dynamique. D'une part, l'agent de système doit explorer son environnement (à savoir des bras de machine) pour recueillir des informations; d'autre part, il doit exploiter les informations collectées pour augmenter la récompense. Comment d'équilibrer adéquatement la phase d'exploration et la phase d'exploitation, c'est une obscurité des problèmes de Bandit, et la plupart des chercheurs se concentrent des efforts sur les stratégies d'équilibration entre l'exploration et l'exploitation. Dans cette dissertation, nous nous concentrons sur l'étude de deux problèmes spécifiques de Bandit: les problèmes de Bandit contextuel et les problèmes de Bandit Multi-objectives.

Des contributions

Cette dissertation propose deux aspects de contributions. La première concerne la classification sous la surveillance partielle, laquelle nous codons comme le problème de Bandit contextuel avec des informations partielles. Ce type des problèmes est abondamment étudié par des chercheurs, en appliquant aux réseaux sociaux ou systèmes de recommandation. Nous proposons une série d'algorithmes sur la base d'algorithme Passive-Aggressive pour résoudre des problèmes de Bandit contextuel. Nous profitons de sa fondations, et montrons que nos algorithmes sont plus simples à mettre en œuvre que les algorithmes en état de l'art. Ils réalisent des biens performances de classification. Pour des problèmes de Bandit Multi-objective (MOMAB), nous proposons une méthode motivée efficace et théoriquement à identifier le front de Pareto entre des bras. En particulier, nous montrons que nous pouvons trouver tous les éléments du front de Pareto avec un budget minimal dans le cadre de PAC borne.

Multi-class PA avec le Bandit contextuel (PAB)

PAB [111] est une variation de PA [33] dans le cas de Bandit contextuel. Étant semblable à l'algorithme PA, la prédiction $\hat{y}_t$ est choisi corrspondant le poids matriciel courant sur des étiquettes $[1,\ldots,K]$. Contrairement au paradigme de l'apprentissage classique, si $\hat{y}_t \neq y_t$, il doit effectuer une exploration. Nous utilisons ϵ-greedy d'échantillonner $\tilde{y}_t$ au lieu de $\hat{y}_t$. Il échantillonne selon une distribution de la probabilité $\mathbb{P}(\tilde{Y}|\hat{y}_t)$

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

Sa mise à jour contient deux items. Le premier item est controllé par l'indicateur $\mathbb{1}(\tilde{y}_t = y_t)$, et il est différent de zéro matriciel uniquement lorsque l'étiquette est prédite correctement. Le deuxième item joue le rôle de lisser le processus d'apprentissage lorsqu'il y a peu de bonne prédiction. Cela signifie que la perte est estimée par un paramètre fixe ρ_c, n'importe quand le processus est aveugle à la bonne étiquette. Lequel est choisi empiriquement.

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

Pour faciliter PAB, son paramètre $\rho_c = 0$. L'experience de l'item $U_{t,1}$ est égal auquel de PA.

Lemma 0.1. *On definit $U_{t,1}$ par la forme $\frac{\mathbb{1}(\tilde{y}_t = y_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t|\hat{y}_t)} U_{PA}(x_t, \hat{y}_t, \tilde{y}_t)$, en sachant que $U_{PA}(x_t, \hat{y}_t, y_t) = \tau_t(\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t))$ dépendant de l'algorithme PA. On peut dire que $\mathbb{E}_{\tilde{Y}}[U_{t,1}] = U_{PA}(x_t, \hat{y}_t, y_t)$.*

La mise à jour de PAB Simple se comporte comme celui de l'algorithme PA, quand $\hat{y}_t = y_t$. Et il fonctionne très bien avec des données linéaires, mais il n'a pas de capacité d'apprendre sur des données réelles et bruyantes. Alors, on y ajoute le item $U_{t,2}$ pour augmenter la stabilité de l'algorithme et réduire la variance de la mise à jour.

Lemma 0.2. $\mathbb{E}_{\tilde{Y}}[U_{t,2}] = 0$.

Lemma 0.3. $\mathbb{E}_{\tilde{Y}}[< U_{t,1}, U_{t,2} >] \leq 0$

PAB fournit un item de la mise à jour dont l'espérance est égale à celle de PA. Il dispose de deux variantes, PAB simple et PAB complète. L'avantage de PAB simple est

sa simplicité et bonne efficace sur l'ensemble des données linéaires et séparables, mais il ne s'adapte pas à traiter de données non-séparables. Par conséquent, nous ajoutons un item anti-interférences, l'espérance duquel est nul. Il fonctionne bien de stabiliser des classifieurs, de réduire la variance de la mise à jour, et performe meilleur que PAB Simple sur des données non-séparables.

Le Bandit contextuel dans la borne de Passive-Aggressive (BPA)

BPA [110] est aussi une variation de PA. Différant d'algorithme de PAB, BPA a une borne d'erreur autant que laquelle de PA. Pour cet algorithme, nous redéfinissons la perte instantanée dans le cas de Bandit:

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

avec $\left(1 - 2\mathbb{1}_{\tilde{y}_t = y_t}\right)$ est -1 quand $\tilde{y}_t = y_t$ et 1 d'autre cas.Cette perte est la standard "Hinge Loss" lorsque la prédiction est correcte: il reste à 0 pour $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \geqslant 1$ puis augmente suivant la valeur diminuée de $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$.

Pour utiliser les outils de l'analyse convexe sous la contrainte ci-dessous,

$$w_{t+1} = \arg\min_{w \in \mathbb{R}^{K \times d}} \frac{1}{2} \parallel w - w_t \parallel^2 \text{ s.t. } l(w;(x_t, y_t)) = 0.$$

Les classifieurs linéaires mettent en jour après chaque essai ,

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

où

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

Theorem 0.1. *Soit* $(x_1, y_1), ..., (x_T, y_T)$ *une suite d'exemples séparables où* $x_t \in \mathbb{R}^d$, $y_t \in \mathscr{Y}$ *et* $\parallel x_t \parallel \leqslant R$ *pour tous t, et* $u \in \mathbb{R}^{K \times d}$. *Alors, l'ensemble de la perte carrée est limitée par,*

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

Theorem 0.2. *Soit* $(x_1, y_1), ..., (x_T, y_T)$ *une suite d'exemples non-séparables où* $x_t \in \mathbb{R}^d$, $y_t \in [1, ..., K]$ *et* $\parallel x_t \parallel \leqslant R$ *pour tous t. Alors, n'importe quel vecteur* $u \in \mathbb{R}^{K \times d}$ *l'ensemble de la perte carrée est limitée par,*

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

Nous reconstruisons " Hinge loss" pour s'adapter au cas de Bandit. L'algorithme BPA est fondée sur cette fonction de perte. Par l'analyse théorique, nous trouvons que l'ensemble de la perte carrée de BPA est limitée par la même borne autant que laquelle de l'algorithme PA qui obtient la rétroaction complète. D'un point de vue empirique, nous avons réalisé des expériences numériques par l'aide de cet algorithme et quelques autres. Lors de présenter une moindre complexité, la performance de notre algorithme est proche de la performance de l'algorithme de Bandit en seconde ordre Perceptron sur tous les types de données. D'ailleurs, cette performance est fréquentement mieux que laquelle des classifieurs supervisés sur certains données non-séparables.

Le Bandit contextuel aux noyaux fonctions

À cette partie, nous proposons deux algorithme en noyau à résoudre des problèmes d'apprentissage en ligne avec rétroaction partielle.

Compte tenu de l'algorithme BPA, sa mise à jour pour RKHS (Reproducing Kernel Hilbert Space) sera présentée comme ci-dessous:

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

où, $\tau_t = \frac{l_t(\mathbf{x}_t, \mathbb{1}_{(\tilde{y}_t = y_t)})}{k(\mathbf{x}_t, \mathbf{x}_t)}$ avec la perte

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

Nous avons besoin de prendre attention à cette perte, qui est différente de la perte de BPA. Il contient un indicateur au lieu d'une paramètre constante, laquelle travaille à maximiser la marge. Notre but est de réduire la complexité et d'éviter certains vecteurs inutiles. Alors que, suite à la croissance des exemples d'apprentissage, nous aurons l'hypothèse $\mathscr{F} = \{f^1, \ldots, f^K\}$:

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

Une autre méthode se réfère à SGD [64]. Comme SGD en espace de RKHS, le but de SGD est de minimiser le risque régularisé:

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

Ici, la perte devrait être remplacée par la perte de l'algorithme BPA. En considérant l'algorithme SGD classique, nous prenons le gradient sur chaque hypothèse f^i de $\mathscr{F}$.

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

où pour $k \in [K]$, $t \in \mathbb{N}$, $f_t^k \in \mathscr{H}$, ∂_{f^k} dénote $\partial / \partial f^k$ et $\eta_t > 0$ est le taux d'apprentissage.

Donc,

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

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

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

Ici, nous proposons certains paramètres en compte $(\sigma_t^1, \ldots, \sigma_t^K)$ avec

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

Par les paramètres σ, la mise à jour pourrait être exprimé en l'équation suivante: pour $\forall k \in \{1, \ldots, K\}$

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

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

Les méthodes **Kernel BPA** et **Kernel SGD avec la perte de Bandit** s'adaptent à la cadre de RKHS. Ces deux algorithmes sont spécials de traiter les problèmes avec la rétroaction de Bandit sur des données non-linéaires. Dans la section 5.3, nous fournissons des détails sur ces approches. La première est une dérivation directe de l'algorithme BPA, qui correspond au produit des deux vecteurs dans RKHS. Cette méthode a une bonne précision, mais la complexité des classifieurs augmente linéairement suivant des exemplaires. Cela apporte des dérangements à l'efficacité. En se référant à SGD [21] et Kernel en ligne [64], nous optimisons les classifieurs au noyau par la perte de Bandit. Bien que sa précision ne parvient pas au niveau de **Kernel BPA**, sa complexité de calcul est stable. Il peut facilement traiter toutes types de données.

Algorithme BPA pour la classification multi-étiquette

Ici, nous introduisons un nouvel algorithme d'apprentissage en ligne, qui est une variante de l'algorithme BPA adapté à la classification multi-étiquette avec le Bandit contextuel.

Dans ce cas, nous prenons la stratégie ϵ-greedy à équilibrer l'exploration et l'exploitation sur l'ensemble des étiquette. À chaque tour $t = 1, 2, \ldots$, l'algorithme sélectionne un sous-ensemble $\tilde{Y}_t$ selon la probabilité $P_t = \left(P(\tilde{y}_t^1 = 1|\hat{y}_t^1), \ldots, P(\tilde{y}_t^K = 1|\hat{y}_t^K)\right)$, avec:

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

Laisser le cardinal de $\hat{Y}$ être noté $Card(\hat{Y}) = \sum_{k=1}^{K} \mathbb{1}_{(\hat{y}^k=1)}$.

Lemma 0.4. *Avec les notations introduites précétement, lorsque l'on sélectionne le sous-ensemble $\tilde{Y}_t$ par la stratégie ϵ-greedy, $\mathbb{E}[Card(\tilde{Y}_t)] = Card(\hat{Y}_t)$.*

La perte instantanée de la classification multi-étiquettes est définie par morceaux $L_t = \sum_{k=1}^{K} l_t^k$, où

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

avec $\beta_t^i = +1$ quand $\tilde{y}_t^i = y_t^i = 1$, -1 quand $\tilde{y}_t^i = 1$ & $y_t^i = 0$ et 0 pour autre cas. Cette perte est le "Hinge loss" standard comme $[1 - \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$ quand la prédiction est positive correcte: il reste à 0 pour $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \geqslant 1$, puis augmente suivant la valeur diminuée de $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$. Au contraire, si la prédiction est positive incorrecte, la perte égal à $[1 + \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$, il reste à 0 pour $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \leq 1$, puis diminue suivant la valeur augmentée de $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$. Pour tous les prédictions négative $\tilde{y}_t^i = 0$, la perte est toujours nulle.

Pour la prédiction, nous sortie une sous-ensemble $\hat{Y}_t \in \{0,1\}^K$ correspondant aux prédictions binaire suivantes:

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

Ensuite, la stratégie ϵ-greedy fonctionne, où $\tilde{Y}_t \in \{0,1\}^K$ est le résultat d'un tirage random sur la distribution $\mathbf{P}_t$ (voit Eq.(5.15)).

Les classifieurs linéaires sont mis à jour à chaque essai en utilisant les outils standards de l'analise convexe. Selon l'optimisation sous contrainte.

$$w_{t+1} = w_t + \tau_t \sum_{k=1}^{K} \left(2\beta_t^k - 1\right)\Phi(x_t, k)$$

où

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

Theorem 0.3. *Soit $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ une suite d'exemples séparables où $x_t \in \mathbb{R}^d$, $Y_t \in \{0,1\}^K$ et $\parallel x_t \parallel \leqslant R$ pour tous t, et $u \in \mathbb{R}^{K\times d}$. Alors, l'ensemble de la perte carrée est limitée par*

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

Theorem 0.4. *Soit* $(x_1,Y_1),\ldots,(x_T,Y_T)$ *une suite d'exemples non-séparables où* $x_t \in \mathbb{R}^d$, $Y_t \in \{0,1\}^K$ *et* $\| x_t \| \leqslant R$ *pour tous t. Alors, n'importe quel matrice* $u \in \mathbb{R}^{K\times d}$,

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

Pour la classification multi-étiquette, nous proposons ce nouveau algorithme. Nous transférons la classification multi-étiquette en plusieurs classifications binaires. En même temps, il apprend des classifieurs multiples par BPA. Cet algorithme, à la fois de l'analyse théorique et résultat empirique, se performe bien.

Un algorithme d'identifier le front d'ϵ-Pareto de MOMAB

Dans cette partie, nous proposons un algorithme pour identifier le front d'ϵ-Pareto de MOMAB. Il étend l'algorithme du naïve (ϵ,δ)-PAC au réglage multi-objective. Ce nouvel algorithme peut identifier les solutions optimales sous une tolérance δ.

Lemma 0.5 (Le borne inférieur)**.** *Si nous ne manquons pas de bras optimale du front d'*ϵ*-Pareto sous une tolérance* δ*, nous avons besoin au moins de tirer* n_{lower} *fois sur chaque bras.*

$$n_{lower} = \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta})$$

Lemma 0.6 (Le borne supérieure)**.** *Si nous pouvons élimiter toutes les bras non-optimales du front d'*ϵ*-Pareto sous une tolérance* δ*, nous devons tirer* n_{upper} *fois sur chaque bras.*

$$n_{upper} = \frac{2K}{\epsilon^2}\ln\frac{2KD}{\delta}$$

Algorithm 0.1 (ϵ-Pareto Identification algorithm)**.**

Initiate parameters ϵ *and* δ
for *Sample each arm* $a \in \mathscr{A}$ $n_1 = \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta})$ *times* ***do***
  $\hat{\mathbf{p}}_a = \frac{1}{n_1}\sum_{t=1}^{n_1} \mathbf{r}_{a,t}$
end for
Identify the set $\mathscr{A}_\epsilon = \{a \in \mathscr{A} | \forall a' \in \mathscr{A}\, a' \not\succ_\epsilon a\}$.
for *Sample each arm* $a \in \mathscr{A}_\epsilon$ $n_2 = \frac{2K}{\epsilon^2}\ln\frac{2KD}{\delta} - \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta})$ *times* ***do***
  $\hat{\mathbf{p}}_a = \frac{1}{n_1+n_2}\sum_{t=1}^{n_1+n_2} \mathbf{r}_{a,t}$
end for
Output $\mathscr{A}_\epsilon^* = \{a \in \mathscr{A}_\epsilon | \nexists a' \in \mathscr{A}_\epsilon, a' \succ_\epsilon a\}$

Par le borne inférieur, l'algorithme peut sortir un ensemble d'ϵ-Pareto qui contient tous les bras optimaux sous une tolérance δ, ensuite nous pouvons éliminer tous les bras non-optimaux après tirer par le borne supérieure. Pour comparer l'ϵ-dominance entre les bras, nous proposons une méthode rapide qui peut mettre en ordre des bras par l'ordre partiel, laquelle est l'amélioration d'algorithme [69] (voit au Chapitre 6).

Les perspectives futures

Il y a certaines limitations de nos résultats à s'adresser dans le futur. En considérant le sujet de la rétroaction de Bandit, nous devons se concentrer des forces sur l'équilibrement entre exploration et exploitation. Nos algorithmes se sont basés sur la stratégie "ϵ-Gloudon" (voit à la section 2.3.4) d'équilibrer entre exploration et exploitation. À chaque tour, il explore des étiquettes par la probabilité fixe $\frac{\epsilon}{K}$, et les exploite par la probabilité $1-\epsilon$. Nous gaspillons le budget, si nous toujours faisons l'exploration par la même probabilité après les classifieurs ont bien joué. Dans les figures 5.9, 5.10 et 5.11, nous observons que les résultats d'algorithme BPA ont moins difference sur les valeurs différentes ϵ. Ce résultat nous suggère naturellement d'utiliser d'ϵ partout les valeurs disponibles, tellement nous pouvons explorer plus aux premières tours, ensuite mettons tous les efforts d'exploiter. À la prochaine étape, nous pouvons essayer le "ϵ-glouton dynamique" [11], le paramètre ϵ duquel attenue correspondant à $1/t$. D'ailleurs, il y a une autre limitation de nos algorithmes: comment de choisir les paramètre, e.g. dans l'algorithme PAB, C pour la frontière douce, ϵ pour "ϵ-glouton"; du KBPA, les paramètre de noyau etc. En générale, c'est nécessite de mettre la méthode "cross-validation" à déterminer des paramètres. Néanmoins, quand il y a deux ou plus de paramètres pour un algorithme, la tâche devient plus difficile.

Le deuxième aspect est sur le problème du front Pareto de MOMAB. En ce moment, nous utilisons l'exploration pure d'identifier les bras optimaux pour sélectionner le front Pareto. Cette méthode peut garantir l'intégrité du front Pareto. Mais, il gaspille plus de budgets que d'autres méthodes. Pour la prochaine étape, nous devons continuer à optimizer cette méthode.

Cette dissertation existe quelques perspectives théoriques et pratiques dans le futur. Du perspective pratique, elle a des prospects vastes. Nous avons mensionné des applications aux réseaux sociaux et au système recommandation. D'autre exemple est prise de décision aux ressources humaines [109]. En se considérant des problèmes comme le routage des paquets Internet ou de distribution d'électricité de réseau intelligent, Bandit en Multi-objective peut proposer une efficace encadrement qui puisse optimiser celles problèmes. L'encadrement de Bandit peut aussi combiner avec d'autre encadrement d'apprentissage

automatique, e.g. “apprentissage profond”, Apprentissage active, apprentissage demi-supervisé, et propose une principale commune pour celle-ci.

Enfin, pendant les années de préparer ma thèse, il non seulement influence ma vue en recherche, mais aussi rectifie mon attitude de vie.

Table of Contents

List of Tables

TABLE **Page**

List of Figures

Part I

General Introduction and State of the art

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 [96], for attempting to compare treatments having unknown effectiveness. This is the prototype of bandit problems.

In 1952, H. Robbins introduced in [86] a problem about sequential selection experiments, that became the foundation of Bandit problems.

In 1957, Richard Bellman wrote a first book [19] on this subject, formulating the Multi-Armed Bandit problem as a class of dynamic programming.

Related to H. Robbins's research, in 1979, Gittins and Jones [55] published a paper to outline an allocation index for sequential experiment problems, and established the stopping rule by a theorem. From then on, more and more proofs of "Gittins Index" have been proposed, i.e. by Whittle [103], Weber [102], Tsitikis [98] etc.

Nowadays, Further researches and applications of Bandit problems have been explored, for example, in fields of Machine Learning by [20, 61, 95], Economy [8, 16, 94] , Cognitive Science [37] etc.

1.1 Motivation

In this section, we present Bandit problems from the following questions:

What are Bandit problems?

Bandit problems, are some classic game problems. They can be imagined as the scenario as follows.

In a casino, a gambler faces to many different slot machines. After pulling one arm, he has a certain probability to earn a reward. Obviously, he does not know the rewarding probability of each arm. To obtain the information of these slot machines, the gambler should pull each arm several times. This process is called “Exploration”. Of course, more explorations he takes, more reliable information he gets. For the gambler, his goal is to earn the largest rewards rather than information. Normally, he should pull the arm with highest estimated rewarding probability according to the current information. It is named “Exploitation”. Bandit problems are how the gambler tells the optimal arm and gets the largest rewards in limited time (or limited budget).

Why to research Bandit problems?

Actually, Bandit problems are not only a game theory, but it also plays a big role in other fields. By solving Bandit problems, we can find the answer to solve other similar problems, such as the following examples.

In the process of clinical trials, doctors often encounter a problem: Treating a same disease, there may be two or even more solutions, but their effectiveness is unknown. This is the prototype of Bandit problems. In 1933, Dr.Thompson proposed “Thompson Sampling” (details in Chapter 2) which is a strategy to identify the most effective treatment, preliminarily solving this problem.

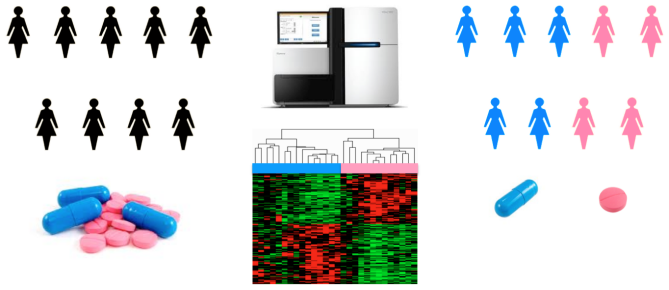

Figure 1.1: Clinical Trials

With the rising development of Internet, Bandit problems have become extremely popular, such as personalized search and Internet advertising service. For these applications, traditional way is to provide a certain number of recommendations by “Collaborative filtering”. Compared with strategies of Bandit problems (refer to Chapter 3), “Collaborative filtering” is unable to work without references in initial stage. However, Bandit strategies are not affected by this problem.

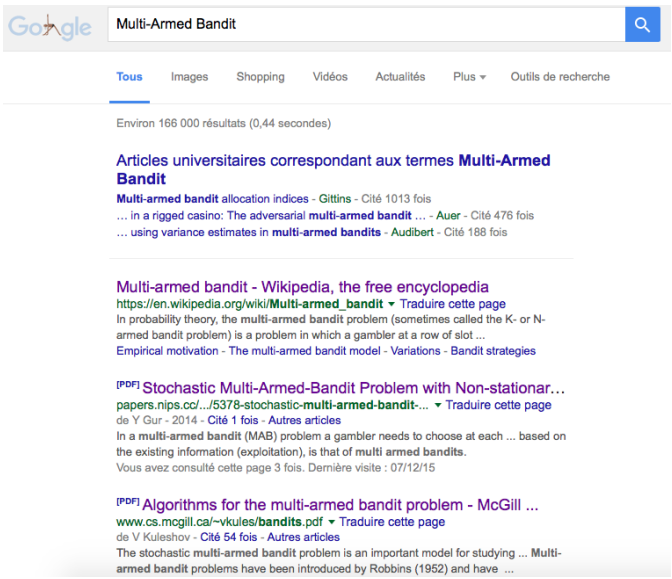

Figure 1.2: Web Search

Otherwise, Google proposed an analytic service [56] based on the Multi-Armed Bandit method (Chapter 2). The purpose of this analytic system is to compare and manage some online experiments. For each experimentation, there maybe a variety of alternative solutions. In the past, “A/B Test” was used to solve this problem, but it takes pure exploration during the exploratory phase, and wastes resources with inferior options [81]. Taking the Bayes

Theorem combined with the Multi-Armed Bandit, Google could reduce the computational time and get a higher accuracy than before.

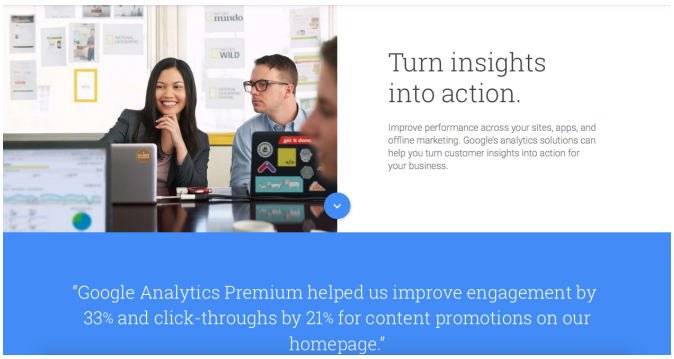

Figure 1.3: Google Analytic Service

Apart from Internet, Bandit problems also have very broad prospects in other domains. During World War II, many countries tried to solve multi-objective multi-tasking attack, and this one could be considered as Multi-Objective Multi-Armed Bandit problem (shown in Chapter 4). Queueing and scheduling [45, 101] is a problem of index policies (refer to Chapter 2), Stock is a Bandit problem in non-stationary environment [50] etc.

How to solve Bandit problems?

The core of Bandit problems is a trade-off between Exploration and Exploitation. Some related strategies and methods will be presented in following Chapters 2, 3, 4, 5, and 6.

1.2 Modeling

From different point of views, Bandit problems can be divided into many categories. They can not be generalized by a uniform modeling. According to the number of trials, Bandit problems can be defined as “Finite horizon” and “Infinite horizon”. The former has a fixed number of trials, the latter is usually unknown in advance. By the number of arms, it can be divided into “Two Arms”, “K arms” and “Many arms”. From the view of system feedback, it can be named “Stationary Bandit”: the distribution of arms is fixed and independant; “Non-Stationary Bandit”: the distribution of arms can be changed during the period of sampling; “Linear Bandit”: all arms are related with each other and satisfy a certain linear function; “Contextual Bandit”: the feedback of arms is associated with side information. Every Bandit problem owns its special model and different strategy. In the next chapters, we will tell about each kind of bandits, their performance, properties and characteristics. Especially, in Chapter 5 and Chapter 6, we propose some novel solutions and algorithms.

Here, we address “Stationary Bandit problem” in order to explain the mechanism of Bandit problems.

Considering Robbins's sequential decision experiment [86], there are several possible choices to be sampled. Generally, operator makes a choice from a set of options at each time. After sampling, operator receives a boolean feedback **1** or **0** as reward. By observing each option's feedback, the fixed but unknown probability distribution of option can be estimated empirically (ex. calculate the rewarding expectation). The goal of Bandit problems is to maximize the received rewards from all steps, minimize the cumulative regret and find the optimal option (or a set of optimal options).

In section 1.1, we have introduced that “Exploration” and “Exploitation” are two core elements to solve Bandit problems. By exploration, operator tries different options to get knowledge which one may be rewarding more frequently. By exploitation, operator repeats to choose an option with better performance. Pursuing the goal of Bandit problems, operator should know when to explore or to exploit. The decisions are always dependent on the potential knowledge of the optimal arms. Performing well on Bandit problems requires trading off between Exploration and Exploitation during all decisions. In early steps, it makes sense to explore, to search for those with the highest reward rates. In the later, it makes sense to exploit those arms which considered to be good, finally maximizing the reward for each current decision.

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

1.3 Thesis Outline

This dissertation consists of 7 chapters by the discussion of Bandit problems.

Chapters 1, 2, 3 and 4, are about the general introduction and the state-of-the-art of Bandit problems.

Chapter 1, introduces the background of Bandit problems.

Chapter 2, tells about Multi-Armed Bandit (simplified as MAB) problem and introduces some probably effective strategies of trading-off between Exploration and Exploitation.

Chapter 3, describes the classification problem under Bandit Feedback. Here, we introduce several algorithms in the state-of-the-art to solve the problems in multi-class/multi-label classification.

Chapter 4, extends Multi-Armed Bandit to the situation where arms are in Multi-Objective (denoted as MOMAB). In this chapter, we introduce some traditional methods to solve MOMAB problems.

Chapter 5 and 6, propose our contributions to Bandit problems.

Chapter 5, proposes some novel algorithms to solve classification with bandit feedback problem, ex. multi-class, multi-label and in Reproducing Kernel Hilbert Space (RKHS). Furthermore, we compare their effectiveness or regret bound by empirical experiments based on some common datasets.

Chapter 6, proposes some optimization methods to find the Pareto front of MOMAB

Chapter 7, is the last part. It makes the conclusions of all chapters in this part and proposes some new directions of research in the future.

Chapter 2

Multi-Armed Bandit

Bandit problem, just as expounded in Chapter 1, was formally proposed in 1952 by H.Robbins. In this chapter, we focus on describing one classical Bandit problems – "Multi-Armed Bandit" (MAB). The purpose of this chapter is to introduce some necessary notions, present some effective strategies, define and analyze some specific issues.

2.1 Environment of Multi-Armed Bandit

The complexity of Multi-Armed Bandit problems, not only stems from the trade-off between Exploration and Exploitation, but also from the wide range of its environments. Each environment of Multi-Armed Bandit provides a different framework. In this section, we address the characteristics of Multi-Armed Bandit in each environment, and take the K-Armed Bandit as the typical case.

2.1.1 Stationary Bandit

Stationary Bandit [86] is the most classical and common bandit problem. The gambler should pick up an arm from the arm setting, and receive a reward as response from the picked arm. The reward obeys a fixed and independent probability distribution.

The K-Armed Bandit in stationary environment, can be defined as following setting. The gambler faces a slot machine with K arms. Each arm k from the arm set $\mathscr{K}$ is characterized by a distribution ν_k with its expected reward μ_k and a variance σ_k^2. All of these parameters are not changeable. In each round $t \geqslant 1$, the gambler selects an arm k_t and receives a sample drawn from ν_{k_t} independently from the past and other arms. Here,

the goal of the gambler is to maximize rewards (minimize regret) or to identify the best arm from all arms by estimating the empirical expected reward after T times pulling.

After T pulls and observations, the gambler samples each arm T_k times (denoted as $T_k = \sum_{t=1}^{T} \mathbb{1}_{k_t=k}$) and estimates the mean of each arm by $\hat{\mu}_k = \frac{1}{T_k} \sum_{t=1}^{T} X_{k_t=k,t}$, where $X_{k_t=k,t}$ denotes the reward when we pull arm k for the t^{th} time. After T observations, we find the optimal arm k^*:

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

Simple Regret In the sequel, $\Delta_k = \mu^* - \hat{\mu}_k$ denotes the gap between the maximal expected reward and the k^{th} arm. The minimal gap can be noted by $\Delta = \min_{k:\Delta_k>0} \Delta_k$. Therefore, the simple regret in the round T equals to 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 by the following formulate.

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

2.1.2 Adversary Bandit

Adversary environment [12] is a challenging problem of Multi-Armed Bandit. In this environment, rewards for each step are selected by an adversary rather than the stationary environment where the rewards are picked from a fixed distribution. Any method of MAB in adversary environment should assume that the information is not symmetric between the gambler and the adversary. Similarly to the stationary environment, we define $\mathscr{K} = \{1, \ldots, K\}$ as a set of arms, and $\mathscr{T} = \{1, 2, \ldots, T\}$ denote the sequence of decision epochs by the gambler. Compared Adversary environment with Stationary environment, the difference is their rewarding distributions. In Adversarial environment, the reward distribution v is changeable at each epoch t under the controlling of an adversary, but the distribution of Stationary case is stable. From another point of view, Adversarial Bandit can be seen as a competition between the gambler and an omniscient adversary.

This issue can be summarized in the following main points:

- the adversary chooses a reward distributions $\mu_{\mathbf{t}} = (\mu_{\mathbf{1,t}}, \ldots, \mu_{\mathbf{K,t}}) \in [\mathbf{0,1}]^{\mathbf{K}}$

- the gambler picks one arm k with no knowledge of the adversary's choice
- the rewards are assigned as $X_{k,t}$, it is drawn i.i.d. by the distribution $\mu_{k,t}$.

Regret of Adversary environment can be presented as:

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

2.1.3 Contextual Bandit

Contextual Bandit [4, 80], is a natural extension of Multi-Armed Bandit. is obtained by associating side information with each arm. Based on this side information, some applications can be naturally modeled as Multi-Armed Bandit problems with context information, also named as Bandit with side information. It is closely related to work on supervised learning and reinforcement learning. In order to facilitate the presentation in Chapter 3, the definitions of Contextual Bandit, will be described with 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 the feature vector $\mathbf{x}_{t,y_t} \in \mathscr{X}$. Where y_t is the optimal arm for x_t, but the gambler does not know this. With the chosen arm $\hat{y}_t$, the gambler receives a reward $r_{(x_t,\hat{y}_t)}$. One should emphasize that the reward is only observed when the arm is chosen.

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

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

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

2.1.4 Linear Bandit

The linear Bandit [1, 27] is also a sequential decision-making problem similar to the other bandit problems, where the gambler has to choose an arm from the arms set at each step t. As a response, the gambler receives a stochastic reward, whose expected value is an unknown linear function of the chosen arm.

Here, we formally present the definition of the linear Bandit problem. On the round t, the gambler is given the arm set $\mathscr{K} \subseteq \mathbb{R}^d$. He selects an arm $k_t \in \mathbb{R}^d$ from the set $\mathscr{K}$. 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 for 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 problem is concerned with sequential decisions. In this section, we focus on the optimal decision process. A MAB problem can be treated as a Markov Decision Process (MDP). However, no approach can scale well on it with the huge of dimensionality.In that situation, Gittins and Jones [53] 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 $X(k) = (x(k,1), x(k,2), \dots)$ presents the state of arm k. Therefore, 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 $\mathscr{G}_k$ is the value of the Gittins index for arm k, two parameters r_k is the reward after pulling arm k and β is a attenuation parameters. The stopping time τ represents the first time at which the index for this arm may be optimal no more. By the way, the decision rule is then to simply choose arm k_t, which can be computed by $k_t = \underset{k \in \mathscr{K}}{\operatorname{argmax}} \mathscr{G}_k(X(k))$.

Theorem 2.1. *The problem posed by a simple family of alternative bandit processes, as setup above, is solved by always continuing the process with the largest* ***Gittins Index****.*

Various proofs of this theorem have been given, the original proof is proposed by Gittins and Jones[53], and a later proof by Gittins [54] relied on an interchange argument. Further simplified by [98, 100]. 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 to select the arm with the largest 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 [89] 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, the difficult point of Bandit problem is to keep a balance between Exploration and Exploitation. That way, it becomes very important to trade off between exploration and exploitation, where 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 the real optimal arms.

The easy way to solve this problem is to select an arm randomly. However, this way mainly relies on luck. So it is not reliable. Another simple way is called "Naive Sampling" approach. It samples each arm by the same number, and then measure the results on comparing their rewards. This approach also has a problem. If the initial number of samples is too small, the confidence is not believable; if the initial number is big enough, it wastes budgets. So we introduce some effective strategies of tradeoff in this section.

2.3.1 Thompson Sampling

Thompson Sampling [96], a randomized algorithm based on Bayesian ideas, is one of the 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, 30].

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 number 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 trial population and the effect could be observed. These observations can 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 random choice is to define the probability that an arm is the best is a Bayesian estimate.

Let θ be a parameter vector to present the choices. The probability of the optimal arm k is

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

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

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 $\theta_1,\ldots,\theta_K \sim P(\mu_1,\ldots,\mu_K|H_t)$.
  Pull arm $k_t = argmax_{k\in\{1,\ldots,K\}}\theta_k$
  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(See in Appendix A.1) There is a question, what is the tradeoff between exploration and exploitation for Thompson Sampling. May [80] tried to separate these 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 expected rewards of an arm could be seen as the sampling drawn from the posteriori

distribution. With these, the exploratory value of an arm can be 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 draws samples with $\theta_k(t) = max(\mathbb{E}[\mu_k], s_k(t))$ 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 sample it. This ensures that the exploratory value of an action is always positive.

2.3.2 Boltzmann Exploration

Boltzmann Exploration (also named Softmax) is based on the axiom of choice [78] and picks each arm with a probability which is proportional of its choices. An arm with larger empirical expected rewards is sampled with a higher probability. Softmax uses a Boltzmann distribution to select the arms. Given the initial empirical means $\hat{\mu}_1(0), \ldots, \hat{\mu}_K(0)$,

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

Softmax depends 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. The choice of parameter τ generally depends on the knowledge of similar action or quantity of the value $\exp \hat{\mu}/\tau$.

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 $P_k(t)$*, 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 at time t.*

$\hat{\mu}_k(t) = \sum_t r_{k,t} / \sum_t \mathbb{1}_{k=k_t,t}$

end for

Softmax can be modified in the same way as the ϵ-greedy strategy in Section 2.3.4 to attenuate the exploration probability, when the temperature τ 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 current round t. The initiation value of τ_0 is set by the user. The decreasing Softmax is analyzed by Cesa-Bianchi [29] with the "Soft-mix" algorithm.

A more complicated variant of the Softmax algorithm, the EXP3 – "exponential weight algorithm for Exploration/Exploitation" (see Appendix A.2) is introduced in [13]. The probability of choosing arm k on round 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 arm i has a reward $r_i(t)$ on round t, $w_i(t+1) = w_i(t)$ 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. A modified version of EXP3, with γ decreasing over time, it is shown by [12], where a regret of $O(\sqrt{KT\log K})$ is achieved.

2.3.3 Upper Confidence Bound

Upper Confidence Bound (UCB) was proposed by Lai [70], to deal with the Exploration and Exploitation dilemma in Multi-Armed Bandit problem by using Upper Confidence Values. Most strategies for trade-off between exploration and exploitation have one weakness: they do not keep track of how much they know about the options. They only pay much more attention to know how much reward they got, i.e. they under-explore options whose initial experiences were not rewarding, even though they don't have enough data to be confident about those options. However, UCB pays attention to not only what it knows, but also how much it knows.

For example, in the stochastic Multi-Armed Bandit problems, the gambler has to choose in trial $t \in \{1,2,\ldots,T\}$ an arm from a given set of arms $\mathcal{K} = \{1,\ldots,K\}$. In each trial t, the gambler obtains a random reward $r_{k,t} \in \{0,1\}$ by choosing the arm k. It is assumed that the random reward $r_{k,t}$ is an i.i.d. random variables with an unknown mean μ_k for arm k.

The goal of the gambler is to get the arm with largest expected reward $\mu^* := max_{k\in\mathcal{K}}\mu_k$. The best expected rewards of this arm $\hat{\mu}^*$ so far serves as a criteria, and other arms are played only if their expected rewards within the confidence interval of $\hat{\mu}^*$. 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 Kullback-Leibler distance between the reward

distributions of the optimal and the suboptimal arm, and $o(1) \to 0$ as $T \to \infty$. This bound was also shown to be asymptotically optimal [70].

Auer [10] introduced the simple, yet efficient UCB algorithm. It is 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 of 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 real 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 real optimal expected 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

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

the 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*

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

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 [14], Auer proposed an improved UCB algorithm (shown in Algorithm 2.3). In this improved model, the 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 [48].

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 [95] (shown in Algorithm 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 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

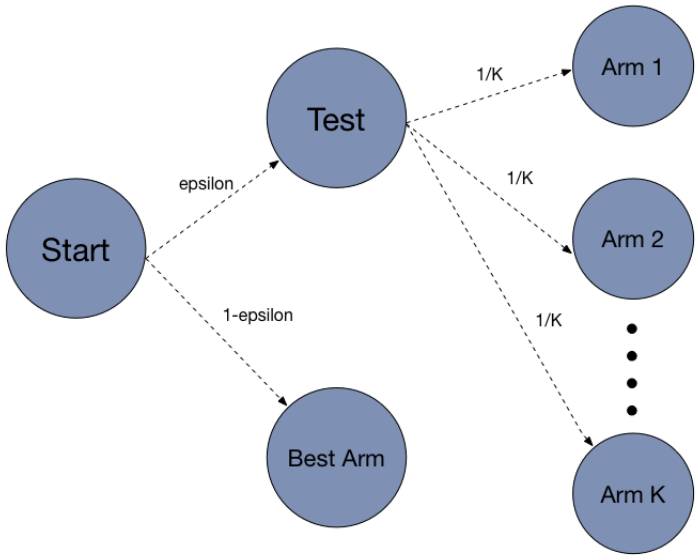

Figure 2.1: the mechanism of epsilon-greedy

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.

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 [11] 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 is less than the number of objects with minimum regret. Yet this bound has a suboptimal dependence on d. In Auer's [11] same paper, it is shown 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 a 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) [97]. 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, $\epsilon(s)$, which is according to the value of error function. 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

Lai and Robbins [70] 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. [63] 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*

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

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

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

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 Equation 2.9 satisfy with equality and the strategy is asymptotically optimal.

2.5 Pure Exploration and Best Armed Identification

Bubeck [24, 25] and Chen [31] proposed to investigate the problem where the gambler may sample arms a given number of times T which may be not necessary known in advance, and be asked to output recommended arm. They advocate to evaluate the performance of a gambler by the simple regret (refer the section 2.1.1). The distinction 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,

Algorithm 2.5 (Pure exploration)**.**

Set rounds T and a arm set K.
for** each arm k=1,2,..., K **do
  for** each round t = 1,2,..., T **do
    Pull arm k and get reward $r_{(k,t)}$ which is drawn i.i.d. randomly
  end for
  $\hat{\mu}_k = \frac{1}{T}\sum_{t=1}^{T} r_{(k,t)}$
end for
Output the optimal arm $k_t^* = \underset{k=1,2,...,K}{argmax}\ \hat{\mu}_k$

The pure exploration problem is about the design of strategies which makes the best use of the limited budget in order to optimize the performance and identify their best choice in a decision-making task. Audibert [9] proposed two algorithms to address this problem: UCB-E and Successive Rejects. The former is a pure exploring strategy based on Upper Confidence Bounds, the latter is a parameter-free method based on progressively 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 up to a logarithmic factor.

UCB-E algorithm

This algorithm is an exploration policy based on Upper Confidence bounds (UCB-E). When the exploration parameter a (shown in Appendix A.3) is taken to be of order $\log T$, the algorithm obviously corresponds to the UCB algorithm with the cumulative regret of order $\log T$. Bubeck[24] has shown that algorithms having at most logarithmic cumulative regret, have at least a (non-cumulative) regret of order $T^{-\gamma}$ for some $\gamma > 0$. Following the demonstration of cumulative regret [9], if UCB-E takes the parameter a of order $\log T$, its probability of error converges to $O(1/T)$ instead of exponential decay.

And taking parameter a in $O(T)$ can explore much more than ever. If UCB-E runs with parameter $a \in]0, \frac{25}{36}\frac{T-K}{H_1}]$, the probability of error is at most of order $\exp^{-a}$ for $a \leqslant \log T$, where 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).

Considering the parameter a, if $a \leqslant \frac{25}{36}\frac{T-K}{H_1}$, it essentially says: the more it explores, the smaller the regret is. Besides, the smallest upper bound on the probability of error is obtained for an order of 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 can 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 other algorithm identifies the best arm in MAB 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 equally pulls all arms which have 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. Here, it denotes H_2 as a quantity $H_2 = \max_{k \in \mathcal{K}} k\Delta_k^{-2}$, the upper error is bounded by $\frac{K(K-1)}{2}\exp\left(-\frac{T-K}{\overline{\log}(K)H_2}\right)$, and it could be proved has a lower bound of sampling times by $\frac{T-K}{4\overline{\log}(K)H_2\Delta_k^2}$.

Chapter 3

Bandit with side information

Contextual Bandit, also named 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. This Classification can be traced back to online supervised classification and Multi-Armed Bandit Reinforcement learning, as Multi-Armed Bandit problem with side information. Langford [71] extended Multi-Armed Bandit setting to the case where some side information is provided. However, this 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 of online learning under the frame of Contextual Bandit, and recall some outstanding researches and contributions.

Classification with Bandit Feedback is composed by two parts: Online learning and Bandit set. Firstly, we formally introduce online learning [90]. Online learning is a fundamental task of machine learning. Different to the batch learning, the dataset of online learning usually comes to be sampled and trained in a sequential order. Otherwise, for some special constraints, e.g. limited memory or partial information, machine learning should take online learning as first choice. In this chapter, Classification with Bandit feedback is the latter.

We here introduce the framework of online supervised learning. Let $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{t-1}, y_{t-1})) \subset \left(\mathbb{R}^d \times \mathscr{Y}\right)^{t-1}$ be an ordered sequence of training instances. Let $\mathscr{F} = (f_1, f_2, \ldots)$ be an ordered sequence of linear mappings. After the previous $t-1$ instances, a mapping $f_{t-1} : \mathbb{R}^d \to \mathbb{R}$, that maps the training space to the reals, has been learned. A new instance $(\mathbf{x}_t, y_t)$ is read from the environment. The learner makes its prediction $\hat{y}_t$ from the values of the mapping $f_{t-1}(\mathbf{x}_t)$. Here, the instance $\mathbf{x}_t$ is not only for testing, but also for training. There

is an algorithm π, which updates the mapping at each new instance: $\pi_{\mathbf{x}_t,y_t} : f_{t-1} \rightarrow f_t$. This completes the process of Online supervised learning.

To estimate the algorithm π, we introduce here some evaluation criterion.

Mistakes: For instance t, if the prediction $\hat{y}_t \neq y_t$, it makes an incorrect prediction. Therefore, mistakes can be accounted as

$$\textbf{mistakes}(\pi, \mathscr{F}) = \sum_{i=1}^{t} \mathbb{1}(\hat{y}_i \neq y_i).$$

The goal of algorithm π is to bound the total number of mistakes, it also calls mistake bounds M. This **mistake bound** can be presented as

$$M = \max_{((\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_t,y_t))\subset(\mathbb{R}^d\times\mathscr{Y})^t} \textbf{mistakes}(\pi, \mathscr{F}).$$

Margin: If the dataset D is separable, there exist a mapping f such that $\forall(\mathbf{x}, y) \in D, \hat{y} = y$. Then the full dataset is separated into different classes. The decision rule used to separate the dataset defines the **boundary** between classes. A useful quantity, called the **margin**, is then the minimal distance between the elements of a class and the class boundary. The objective of many learning algorithms is thus to maximize this margin by designing the appropriate class boundaries (e.g. SVM, PA algorithm, Weighted Perceptron).

Loss, also named as instantaneous loss, indicates the penalty for an incorrect prediction, let $l_t(\mathbf{x}_t, \hat{y}_t, y_t)$ be the loss for prediction $\hat{y}_t$ instead of y_t. For all losses of past instances, it called **cumulative loss**, denoted as $L_t = \sum_{i=1}^{t} l_i(\mathbf{x}_i, \hat{y}_i, y_i)$.

For loss function, there are varieties of expressions, e.g. zero-one loss, non-symmetric loss, squared loss and hinge loss. Here, we introduce binary hinge loss [17, 51] as an example. For a linear mapping f, the max margin for datasets is γ. The **hinge loss** in the binary case (i.e. $y \in \{-1, 1\}$) for an instance $(\mathbf{x}_t, y_t)$ is :

$$l_f(\mathbf{x}_t, \hat{y}_t, y_t) = \textbf{max}\left(0, \gamma - y_t.f(\mathbf{x}_t)\right) = [\gamma - y_t.f(\mathbf{x}_t)]_+$$

Regret of an online learning algorithm presents the difference between the loss of classifier f_t and the optimal classifier f^* who is a classifier with largest margin.

$$R(\mathscr{F}, T) = \sum_{t=1}^{T} l(\mathbf{x}_t, f_t(\mathbf{x}_t), y_t) - \sum_{t=1}^{T} l(\mathbf{x}_t, f^*(\mathbf{x}_t), y_t).$$

Regret bound measures the performance of an online algorithm relative to the performance of optimal classifier.

Sometimes, datasets are non-linearly separable. So, Vapnik [99]proposes an explicit mapping **Kernel function** to get linear learning algorithms to learn a nonlinear function

or decision boundary. For two variables $\mathbf{x}$ and $\mathbf{x}'$ from the input space $\mathscr{X}$, let certain functions $K(\mathbf{x},\mathbf{x}')$ be an inner product in another space $\mathscr{X}^\dagger$. For simplicity understand, this kernel mapping can be written in the form of a feature mapping $\phi : \mathscr{X} \to \mathscr{X}^\dagger$ which satifies

$$K(\mathbf{x},\mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

, where $\mathbf{x}$ and $\mathbf{x}'$ are primal variables, and $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$ are dual variables. The dual space is also named Reproducing Kernel Hilbert Space (RKHS). More details are given in section 5.3.

In the next sections, we focus on introducing some supervised classification algorithms in state-of-the-art. From the side of Bandit feedback, we present some important classification algorithms working with Bandit Feedback, most of them being based on the supervised learning algorithm combining some bandit strategies. After completing the description of the Multi-class classification with Bandit Feedback, we pose a novel problem, the Multi-Labels classification working with partial feedback. Then, we provide some analysis and an effective algorithm on this issue.

3.1 Multi-class 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 $\mathscr{Y} = \{1,\dots,K\}$, with $K \geq 2$. In this protocol, forecaster keeps classifiers parameterized $w = (w_1, w_2, \dots, w_K)$ from the hypothesis space $\mathscr{W} \subseteq \mathbb{R}^{K\times d}$. At each steps $t = 1,2,\dots,T$, the side information $x_t \in \mathscr{X}$ is sampled i.i.d. at random, then forecaster predicts the label $\hat{y}_t$, by the linear hypothesis w_t:

$$\hat{y}_t = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}} < w_t, \Phi(\mathbf{x_t}, k) > \tag{3.1}$$

In the standard online protocol, the 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, the forecaster only observes an indicator $\mathbb{1}(\hat{y}_t = y_t)$, that is whether the prediction at time t is correct or not. With Bandit feedback, its cumulative loss is defined as the following format. And the one with Bandit Feedback is defined as below:

$$L_{BF} = \sum_{t=1}^{T} l_{BF}(w_t, \mathbf{x}_t, \mathbb{1}(\hat{y}_t = y_t)) \tag{3.2}$$

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 aim at learning a predictor $h : \mathscr{X} \rightarrow \mathscr{Y}$, 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 often mentioned: One-versus-All and All-versus-All. See Figure 3.1, consider some points classified into three classes (classified as their colors), by the method One vs All, it is 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 [35], where Perceptron [87] and Winnow [75] 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 [65]. The goal of all classification algorithms, is to minimize the mistake bound M or regret R. To achieve this goal, the algorithms update the classifiers at each trial.

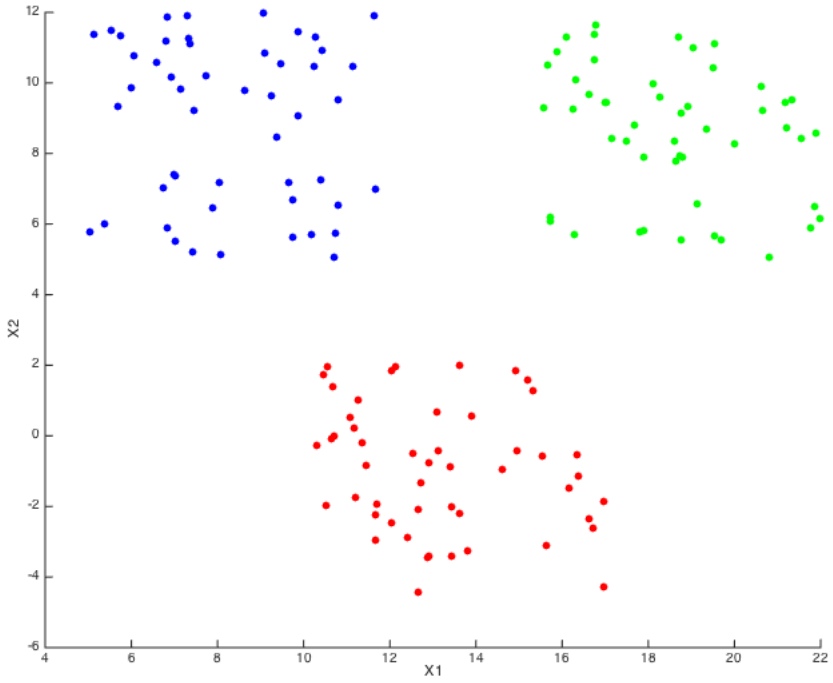

Figure 3.1: Multiclass task

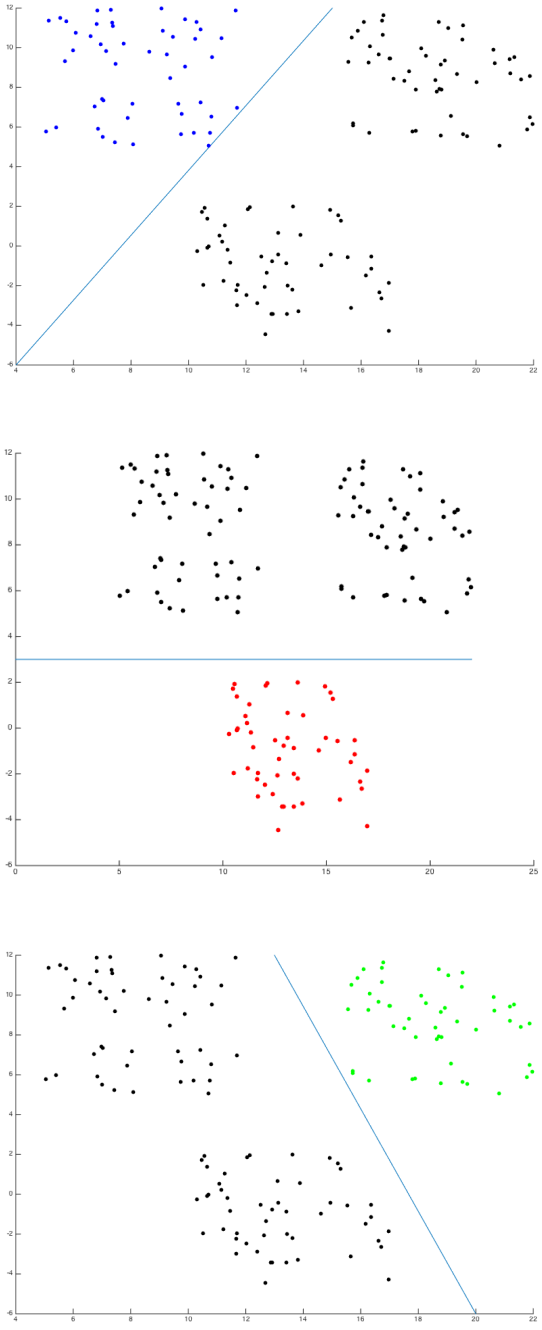

Figure 3.2: Reduction from multiclass classification to binary classification (One vs All)

Here, we make some definitions for simplifying the writing: the feature vector represen-

tation $\Phi(\mathbf{x}, y)$ 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 columns of feature size d. All columns but the y'th column of $\Phi(\mathbf{x}, y)$ are set to the zero vector while the y'th column is set to $\mathbf{x}$. A multi-class algorithm produces a hypothesis $w \in \mathbb{R}^{K \times d}$ from $\mathscr{W}$ on every online round. Like for the construction of $\Phi(\mathbf{x}, y)$, w is composed of K columns of size d and we denote column r by w_r. By construction, we set $< w, \Phi(x_t, r) >=< w_r, x_t >$.

The binary hinge loss has been introduced at the head of this chapter. We here present a variant of this loss function in Multi-class frame.

$$l(w_t;(\mathbf{x}_t, y_t, \hat{y}_t)) = [\gamma + < w_t, \Phi(\mathbf{x}_t, \hat{y}_t) > - < w_t, \Phi(\mathbf{x}_t, y_t) >]_+, \tag{3.3}$$

where parameter γ is the margin of this algorithm. The hinge loss is a convex function. It is not differentiable, but has a subgradient with respect to different classifiers. So many quadratic optimization can work with it, e.g. algorithm SVM. For binary hinge loss, its subgradient to classifier w can be represented as:

$$\frac{\partial l_t(w, \mathbf{x}_t, y_t)}{\partial w} = \begin{cases} -\hat{y}_t \cdot \mathbf{x}_t & \text{if } \hat{y}_t \cdot y_t < \gamma \\ 0 & \text{otherwise} \end{cases}$$

And the subgradient for the Multi-class hinge loss:

$$\frac{\partial l_t(w, \mathbf{x}_t, y_t)}{\partial w} = \begin{cases} \Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t) & \text{if } l_t > 0 \\ 0 & \text{otherwise} \end{cases}$$

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

Perceptron was proposed by F. Rosenblatt [87]. It is a general computational model with some numerical weights. By Minsky and Papert [83], 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 Perceptron algorithm looks 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, the initial weights vector w_0 is chosen randomly. For the binary state, if the instance on round t^{th} x_t belongs to the set P, the class y_t 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

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

The Perceptron in Multi-class Classification with K classes looks 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, with non-separable datasets, there is no soft margin. Perceptron updates according to the incorrect classification for those vague instances. In that case, it will affect the classification result.

$$w_{t+1} = \begin{cases} w_t & \text{if } \langle w_t, \Phi(x_t, y_t) \rangle > 0 \\ w_t + \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) & \text{elsewhere} \end{cases}$$

Its mistake bound convergence in $(R/\gamma)^2$ where $R = \max \parallel x_t \parallel$ and $\gamma = \min| < u, x_t > |$ with optimal classifier u. More details of Perceptron see Appendix A.5.

Second-order Perceptron[28]. In the previous part, we talked about a popular, local and greedy linear algorithm– Perceptron. Instead of Perceptron, a second order variant of Perceptron proposed by Cesa-Bianchi sets a correlation matrix S_t, where $S_t = \sum_{s=1}^{t} x_s x_s^T \in \mathbb{R}^{d \times d}$ is positive, there exists a matrix $S_t^{-1}, s.t. S_t^{-1} S_t = I_t$, I_t is an identity matrix. This matrix can measure the correlation of instances by a quadratic form $\mathbf{x}_t^T M^{-1} \mathbf{x}_t$, i.e. if the quadratic tends to quite small, they are correlated tightly.

In the basic form, Second-order Perceptron algorithm takes an input parameter $a > 0$. To compute its prediction in trial t the algorithm uses a matrix $S_{t-1} \in \mathbb{R}^{d \times d}$ and a classifier $w_{t-1} \in \mathbb{R}^{K \times d}$, where subscript $t-1$ indicates the number of updates. 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 as $\hat{y}_t$. 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 large scale 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 [41] can maintain each feature with a different confidence level, for the features with low degree of confidence need to update more than the one with high level.

Passive-Aggressive Algorithm[33] is an effective framework for performing max-margin online learning. Here, we address Online Passive-Aggressive algorithms, who learn the classifiers from the linear hypothesis space with a set of ordered instances.

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

$$e(w_t;(x_t,y_t)) = < w_t, \Phi(x_t,y_t) > - \underset{s \neq y_t}{max} < w_t \cdot \Phi(x_t,s) > .$$

The margin is positive only if the product of the relevant label $< w_t, \Phi(x_t,y_t) >$ is bigger than the one of other irrelevant labels. This definition of margin computes an instantaneous loss by hinge-loss function as follows,

$$l(w;(x,y)) = \begin{cases} 0 & e(w;(x,y)) \geqslant \gamma \\ 1 - e(w;(x,y)) & \text{otherwise} \end{cases} \tag{3.4}$$

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

$$w_{t+1} = \underset{w \in \mathbb{R}^{K \times d}}{\operatorname{argmin}} \frac{1}{2} \parallel w - w_t \parallel^2 \quad s.t. \;\; l(w;(x_t,y_t)) = 0. \tag{3.5}$$

Intuitively, if w_t suffers no loss from the new instance, i.e. 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} satisfying that $l_t(w_{t+1};(\mathbf{x}_t,y_t))$ attains no loss.

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

$$w_{t+1} = \underset{w}{\operatorname{argmin}} \frac{1}{2} \parallel w - w_t \parallel^2 \quad s.t. \;\; \langle w, \Phi(x_t,y_t) - \Phi(x_t,s_t) \rangle \geqslant \gamma \tag{3.6}$$

The apparent benefit of this simplification lies in the fact that Eq. 3.6 has a closed form solution. To draw the connection between the multiclass and binary classification, considering 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. 3.6 is

$$w_{t+1} = w_t + \tau_t (\Phi(x_t,y_t) - \Phi(x_t,\hat{y}_t)) \tag{3.7}$$

with

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

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,

$$l_t = l(w_t;(x_t,y_t)) = [\langle w_t, \Phi(x_t,y_t) - \Phi(x_t,\hat{y}_t) \rangle + \gamma]_+$$

For the bounds of multiclass PA algorithm, one needs to cast the assumption that for all t, $\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \| \leqslant R$ holds. 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 Algorithms for Multi-class 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. Sometimes, the environment just provides a partial feedback instead of a full one. Such problems are a natural extension of the multiclass problems, called the bandit versions of multiclass prediction problems.

Here, K denotes the number of classes, and $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ denotes 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, $\hat{y}_t \in \{1, \ldots, K\}$ denotes the prediction. Unlike the classical online learning setup where an oracle provides the true class label $y_i \in \{1, \ldots, K\}$ to the learner, the learner only receives a one bit response telling whether the prediction equals the true label, i.e., $\mathbb{1}[y_t = \hat{y}_t]$. The Bandit feedback is a special case of a Contextual Bandit with side information. So the goal of this problem is not only to minimize the error bound, but also to keep the 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 need to 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, being optimal or sub-optimal depending on the hypothesis. Unlike the supervised learning case, we have no knowledge about the true label y_t. So, it is necessary to sample other labels to explore more label information.

In this section, we introduce few traditional multiclass classification approaches combined with Bandit policies.

Banditron [62] (see in Appendix A.8), is a simple but effective learning strategy for online classification with bandit feedback, which is based on the Perceptron algorithm. 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 [87]).

Similar to the Perceptron, at each round, the prediction $\hat{y}_t$ can be the best label according to the current weight matrix w_t, i.e. $\hat{y}_t = \underset{i \in \{1,\ldots,K\}}{\operatorname{argmax}} < w_t, \Phi(x_t, i) >$. 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).

With the randomized prediction $\tilde{y}_t$, the above intuitive argument formalize the update matrix $\tilde{U}_t$.

$$\tilde{U}_t = \frac{\mathbb{1}(\tilde{y}_t = y_t)}{P(\tilde{Y} = \tilde{y}_t|\hat{y}_t)}\Phi(\mathbf{x}_t, \tilde{y}_t) - \Phi(\mathbf{x}_t, \hat{y}_t)$$

where $P(\tilde{Y} = \tilde{y}_t|\hat{y}_t)$ is the probability of predicting $\tilde{y}_t$, it is according to $\hat{y}_t$. 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[62] show that the expected value of the Banditron's update matrix $\tilde{U}_t$ is exactly the Perceptron's update matrix U_t. Banditron is a linear predictor with ϵ-Greedy strategy with Bandit feedback. Its mistake bound is bounded by $O(T^{2/3})$.

Confidit[34] proposes a different strategy of exploration. It uses the same principles as 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 the score values, but also a non-negative uncertainty values for these scores, denotes by $\epsilon_{i,t}$, which is 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 necessity to obtain information about it. Specifically, this algorithm is based on the Second Order Perceptron (see section 3.1.1), i.e. 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 on this algorithm is given in Appendix A.9. Confidit develops the Second Order Perceptron with UCB strategy to solve the classification problem

with Bandit feedback. And it uses the correlation matrix to estimate the uncertainty of label set. Its regret bound is of $O(\sqrt{T}\log T)$, which is better than the one of Banditron.

3.2 Multi-label Classification with Bandit feedback

After introducing the problem of Multi-class Classification with Bandit feedback, we here address another problem of Bandit feedback: Multi-label Classification with Bandit feedback. This problem exists in many fields, for example in recommender systems. Generally, the number of locations is limited on a web-page to show ads. The goal of a recommender system is to find the user's favorite ads, and push them on the web-page. Like other Bandit problems, the system can not observe the user's favorites ads directly. Actually, during the phase of exploration, the system can obtain some knowledge about the user by a Bandit feedback.

Learning proceeds in rounds: at each step t, the system receives an instance $\mathbf{x}_t$ and outputs a subset $\hat{Y}_t$ of labels from a finite set of all possible labels $\mathscr{Y} = \{1,2,\ldots,K\}$. The restriction of the size of $\hat{Y}_t$ corresponds to the limited locations of web-page or other conditions. The system intends to find the true set associated with $\mathbf{x}_t$. However, it can never observe the true label set. Instead of the true label set, the system can receive a feedback from the user: $Y_t \cap \hat{Y}_t$, where $Y_t \subseteq \mathscr{Y}$ is a label set associated with $\mathbf{x}_t$. In the restricted case $|\hat{Y}_t| = 1$ for all t, it becomes a familiar problem: Multi-class Classification with bandit feedback.

3.2.1 Multilabel Classification

Different to Multi-class Classification, Multi-label Classification is more complicated. This section first introduces the supervised learning Multi-label Classification. Multi-class Classification has been introduced in section 3.1, the goal of this problem is to learn from a set of instances associated with unique label $y_t \in \mathscr{Y}$. Unlike this problem, Multi-label Classification is to learn from a set of instances where each instance belongs to a set of classes $Y_t \subseteq \mathscr{Y}$ where $|Y_t| \geqslant 1$.

Here we define some notations of Multi-label Classification. Let $\mathscr{X}$ be an instance space, and $\mathscr{Y}$ be a finite set of class labels. An instance $\mathbf{x}_t \in \mathscr{X}$ is represented in terms of features vector $\mathbf{x}_t = (x^1,\ldots,x^d) \in \mathbb{R}^d$. A subset Y_t, associated to the instance $\mathbf{x}_t$, can be denoted as a binary vector $Y_t = (y_t^1,\ldots,y_t^K)$, where $y_t^i = 1$ if and only if label y_t^i is associated to the instance $\mathbf{x}_t$.

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

Refer to [93], it presents a number of basic transformation methods. Some of them transform Multi-label problem into multiple Multi-class problem, some address this problem by ranking.

Here, we take some methods to explain how to solve Multi-label problem. To describe more easily the methods, we use an example multi-label data in Table 3.1. There are four instances that belong to at least one of 4 classes $\{1,2,3,4\}$.

Table 3.1: Example of multi-label dataset

Instance	Label Set
x_1	$\{1,4\}$
x_2	$\{2,3,4\}$
x_3	$\{1,3\}$
x_4	$\{2,4\}$

Copy transformation this method replaces each example $(\mathbf{x}_t, Y_t)$ with $|Y_t|$ copies $(\mathbf{x}_t, y_t^k)$, for each label $y_t^k \in Y_t$. So, instance $(x_1, \{1,4\})$ represented as $(x_1, \{1\})$ and $(x_1, \{4\})$. There is an extension to this, it is to use a weight of $\frac{1}{|Y_t|}$ to each created examples.

Ignore transformation simply ignores the multi-label of each instance, only left single label for running. For each instance, labels will be selected depending on use reference. There can be several versions: select-min (least frequent), select-max (most frequent), and select-random (randomly selected).

Binary Relevance (BR) is one of the most popular transformation methods. It first creates K datasets ($K = |\mathscr{Y}|$), each for one label, and trains a classifier on each of these datasets. All new datasets contain the same number of instances as the original data, but each dataset D_k, $1 \leqslant k \leqslant K$ positively labels instances that belong to class y^k and negative otherwise. Table 3.2 shows the example dataset for BR.

Once these datasets are ready, it is easy to train a binary classifier for each. For any new instance, BR outputs a set of the labels $\hat{Y}_t$ that are predicted by K binary classifiers.

Calibrated Label Ranking (CLR) proposed by Furnkranz[77]. He argues that providing a relative order of the labels though ranking does not have a natural "zero-point" and therefore, does not provide any information about the absolute preference. CLR can

Table 3.2: Transformed data produced by Binary Relevance (BR) method

Instance	Label 1
x_1	1
x_2	−1
x_3	1
x_4	−1

Instance	Label 2
x_1	−1
x_2	1
x_3	−1
x_4	1

Instance	Label 3
x_1	−1
x_2	1
x_3	1
x_4	−1

Instance	Label 4
x_1	1
x_2	1
x_3	−1
x_4	1

distinguish all labels between relevant and non-relevant classes on label ranking. Introducing an additional label to the original label set, which can be interpreted as a "zero-point" (between relevant and non-relevant). Thus a calibrated ranking,

$$y_t^{\sigma(1)} \succ y_t^{\sigma(2)} \succ \cdots \succ y_t^{\sigma(j)} \succ y^0 \succ y_t^{\sigma(j+1)} \succ \cdots \succ y_t^{\sigma(K)}$$

Clearly, this ranking of labels (ignore the "zero-point" label y^0) creates a bipartition of relevant set ($y_t^{\sigma(1)} \dots y_t^{\sigma(j)}$) and irrelevant set ($y_t^{\sigma(j+1)} \dots y_t^{\sigma(K)}$). Then, each example can be annotated by a calibration label. For the labels of relevant set, it could be treated as a positive example, for the labels of non-relevant set, it could be treated negatively. Thus, it can be treated by some binary relevance methods.

3.2.2 Algorithm for Multi-label Classification with Bandit feedback

There are so many literature to introduce Multi-label Classification [49, 67, 79, 93]. Such problems are collectively referred to the supervised learning. As opposed to the supervised case, in this section, we introduce an algorithm proposed by Gentile [32], who firstly propose a framework to solve the problem of Multi-label Classification with Bandit feedback.

Algorithm of Multi-label Classification with Bandit feedback

This algorithm is based on 2nd-order descent method (shown in Appendix A.10). It uses a linear predictor with a cost-sensitive Multi-label loss. This loss is generalized from the standard Hamming loss, and can be taken into account the distance between Y_t and $\hat{Y}_t$. Otherwise, it proposes a cost function $C(\hat{Y}_t)$ to calculate the cost for the prediction set $\hat{Y}_t$. Associated with $\hat{Y}_t$, it possibly takes into account the order in which i occurs within $\hat{Y}_t$. Specifically, given a constant parameter $a \in [0,1]$ and costs $c = \{c(i,s), i = 1,\dots,s, s \in [K]\}$, 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 this cost-sensitive Multi-label loss function, it can output the prediction set $\hat{Y}_t$ to be a ranked label list. Due to the Bandit feedback, only the relevance labels can be observed, i.e. $Y_t \cap \hat{Y}_t$. So the ranking is restricted to the deemed relevant labels only, and to provide a no supervised ranking information within this set. Furthermore, this loss function provides an effective method to explore the predictions.

Let $\mathbb{P}_t(\cdot)$ be a shorthand for the conditional probability $\mathbb{P}_t(\cdot|\mathbf{x}_t)$. This marginal function $\mathbb{P}_t(y_t^i = 1)$ can present the Bayes optimal ranking (see in [40]). It satisfies,

$$\mathbb{P}_t(y_t^i = 1) = \frac{g(-\mathbf{u}_i^T \mathbf{x}_t)}{g(\mathbf{u}_i^T \mathbf{x}_t) + g(-\mathbf{u}_i^T \mathbf{x}_t)}, i = 1, \ldots, K \tag{3.8}$$

for K vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^d$ and functions $g : D \subseteq \mathbb{R} \to \mathbb{R}^+$. The model is well defined if $\mathbf{u}_i^T \mathbf{x}_t \in D$ for all i and all $\mathbf{x}_t \in \mathbb{R}^d$. For the sake of simplicity, it assumes that $\| \mathbf{x}_t \| = 1$ for all t.

This algorithm proposes a novel problem: Multi-label Classification with Bandit feedback. It generalizes a linear hypothesis relied on UCB to solve this problem. It formalizes a method to set apart the Multi-label loss from the supervised settings. Thanks to the cost-sensitive multi-label loss, it can split the labels sets into relevant and no-relevant sets. This operation is totally automatic, and bounded regret by $O(T^{1/2} \log T)$.

Chapter 4

Multi-Objective Multi-Armed Bandit

Multi-Armed Bandit is a machine learning paradigm used to study and analyze resource allocation in stochastic and uncertain environment. We have introduced this game theory in Chapter 2. Let's consider a special case that the slot machine of MAB responses a vector instead of scalar as reward. Furthermore, every element of this vector corresponds to a certain distribution. Then, MAB problem becomes a Multi-Objective Optimization problem (MOO) in Bandit environment, this kind problem is denoted as Multi-Objective Multi-Armed Bandit (MOMAB). In this chapter, we firstly introduce the MOO problem, after that, we will introduce several concepts and some solutions of MOMAB problem.

4.1 Multi-Objective Optimization

A simultaneous optimization of two or more conflicting objectives is called Multi-Objective Optimization(MOO). This kind problem is much more complicated than problems of single objective optimization, however many practical optimization problems, especially some engineering design optimization problems, exist naturally in the form of MOO. For example, designing a supply chain should consider many factors: time cost, economic cost, human resource and other possible constrains. From one aspect to optimize, it may cause some waste of other aspects. As a result, MOO problems usually require a collaborative optimization of multiple aspects. It can be described as the following mathematical model.

A MOO problem has a variable espace $\mathscr{X} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ and an objective espace $\mathscr{Y}$ with a mapping $\mathbf{f}: \mathscr{X} \to \mathbb{R}^d$. Here, we optimize $\mathbf{f}$ with each objective f_i to be maximized,

$i \in \{1,\ldots,d\}$.

This optimization problem is formulated as follows:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ldots,f_d(\mathbf{x})) \text{ s.t. } \{\max f_1(\mathbf{x}),\ldots,\max f_d(\mathbf{x})\} \tag{4.1}$$

where a variable vector $\mathbf{x}$ ranges in the variable space $\mathscr{X}$, $\mathbf{y}$ is an objective vector with a mapping $\mathbf{f} = \{f_1,\ldots,f_d\}$ where f_i mapping $\mathscr{X}$ onto $\mathbb{R}$. The objective function is the mapping $\mathbf{f}: \mathscr{X} \to \mathbb{R}^d$.

4.1.1 Front Pareto setting

To solve an MOO problem, the key is to trade the conflict off between all objectives f_i with $i = 1,2,\ldots,d$. This subsection concentrates on some concepts of "Pareto optimization", which is considered as the core of MOO. Pareto optimization is proposed by the engineer and economist Vilfredo Pareto[84]. It states that:

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

Several definitions related to Pareto optimization are frequently used in MOO literature (referring to [15, 38, 39, 46, 72]) as the following.

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 $\mathbf{y}'$(denoted $\mathbf{y} \succeq \mathbf{y}'$) iff $y_i \geqslant y'_i, \forall i \in [1,\ldots,d]$.

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

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

Definition 4.4. Pareto optimality The solution $\mathbf{x}^*$ is Pareto optimality iff its correspondent objective vector $\mathbf{f}(\mathbf{x}^*)$ does not dominate by any other objective vector $\mathbf{f}(\mathbf{x})$, that $\mathbf{x} \nexists \mathscr{X}$ s.t. $\mathbf{f}(\mathbf{x}) \succ \mathbf{f}(\mathbf{x}^*)$.

Definition 4.5. Pareto front Given a set $P \subseteq \mathscr{X}$, P^* is a subset of P. Any element of the set P^* is not dominated by elements of P, referred to as Pareto front w.r.t. P.

$$P^* = \{\mathbf{x} \in P : \nexists \mathbf{x}' \in P \text{ s.t. } \mathbf{f}(\mathbf{x}') \succ \mathbf{f}(\mathbf{x})\}$$

Definition 4.6. Comparison between non-dominated sets A non-dominated set P_1 is said to be better than another non-dominated set P_2 (denoted $P_1 \succ P_2$) iff every $\mathbf{x} \in P_2$ is weakly dominated by at least one $\mathbf{x}' \in P_1$ and $P_1 \neq P_2$.

Definition 4.7. Pareto rank There is a set of variables P with their objective vectors $\mathbf{f}(\mathbf{x}) \subseteq \mathbb{R}^d$ where $\mathbf{x} \in \mathscr{X}$. Pareto rank is determined by an iterative manner as following. All non-dominated variables of P (denoted as $\mathscr{F}_1(P)$) are given rank 1. Then this set $\mathscr{F}_1(P)$ will be removed from set P. From the set of all rest variables, to find a set of non-dominated variables of $P/\mathscr{F}_1(P)$, this set is given rank 2 (denoted as $\mathscr{F}_2(P)$). This iterative action stops until all variables of P have their rank.

By Pareto rank, we can directly and easily compare two variables sets, i.e. $\mathscr{F}_i(P) \succ \mathscr{F}_j(P)$ if $i < j$. However, how to identify the Pareto front is the most difficult problem of MOO. Normally, this problem is solved from two aspects, one is from its definition as the starting point [39, 112], through comparing dominance relationship to find the elements of Pareto Front. Another is to aggregate Multi-Objective problem to Single-Objective problem [82, 91, 105]. The former can traverse all optimal solutions of Pareto front, but it has higher complexity. While the latter is more effective to find an optimal solution, unfortunately it is difficulty to traverse all optimal solutions. In the next sections, we will introduce some methods to identify the Pareto front of MOO from these two aspects.

4.1.2 Dominance method

Dominance method is a way to identify Pareto front of MOO, it is started with the definition of Pareto front (see the Definition 4.5). Here, we quote it again by the following mathematical model.

Let $\mathbf{x}$ be a variable from the variable espace $\mathscr{X}$ with a mapping $\mathbf{f}: \mathscr{X} \to \mathbb{R}^d$. The Pareto front is a set P^* of all variables $\mathbf{x}^* \in \mathscr{X}$, where $\nexists \mathbf{x}$ s.t. $\forall i \in [1,\ldots,d]\ f_i(\mathbf{x}) > f_i(\mathbf{x}^*)$.

Global Simple Evolutionary Multi-objective Optimization (Global SEMO) [59], is based on Evolutionary Algorithms to solve the problems of Multi-Objective Optimization. Global SEMO creates a set P of non-dominated variables. For the initialization of this set, it is drawn uniformly at random from the variables espace. At each iterative time, a new variable $\mathbf{x}$ is drawn uniformly at random from the set of $\mathscr{X}$. An offspring $\mathbf{y}$ is created by applying a mutation operator to $\mathbf{x}$. Resorting to the global mutation operator which flips each bit of $\mathbf{x}$ with probability $1/n$. The offspring is added to the set P if it is not dominated by any other variables of P. Some variables will be deleted from the set P, if they are

weakly dominated by y. This iterative action will continue until all variavles of set $\mathscr{X}$ have been drawn., and output the set P to be Pareto front. (see in Appendix A.11)

For theoretical statistic, the number of all rounds is called the runtime of algorithm. The expected runtime refers to the expectation of its random variables. The run-time of this algorithm has been analyzed by [23, 52], performs as well as classical combination optimization problems.

Global Diversity Evolutionary Multi-objective Optimizer (Global DEMO$_\epsilon$) (shown in Appendix A.12) combines algorithm Global SEMO and the concept of ϵ-dominance [59]. The principle of Global DEMO$_\epsilon$ is to partition all variables into several area with ϵ-dominated relationship. It measure each variables of ϵ-dominated area by an index vector. Then, it runs the same process of algorithm Global SEMO, and compares the index value instead of each objective of variable.

This method has two important properties. The first one is to identify an ϵ-Pareto front. This is useful for some MOO problems under an unstable environment. And the second one is its size of evolutionary generation is bounded.

4.1.3 Aggregation method

Besides the dominance method, another useful method is to aggregate the Multi-objective into a Single-objective. Scalarization is the most common way to solve this problem. However, a single-objective environment only results in a single target. To generate all elements of Pareto front, we need a set of scalarization functions to run. In this case, it exposes its shortcomings that it is blindness to search all optimal solutions. Of course, its advantage is obvious that it can quickly find a certain solution with special weight. There are several types of scalarization functions. We here introduce three scalarization function.

The linear Scalarization is the most popular scalarization function due to its simplicity. It weights each value of the objective vector and the result is the sum of these weighted values. The optimal target is to maximize the linear scalarization:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathscr{X}} f(\mathbf{x}) = \arg\max_{\mathbf{x}\in\mathscr{X}} \omega^1\mathbf{x}^1 + \cdots + \omega^d\mathbf{x}^d$$

where $(\omega^1,\ldots,\omega^d)$ is a set of predefined weights with reference and $\sum_{j=1}^{d}\omega^j = 1$. Here, we point out its weakness is unable to solve a non-convex Pareto front.

The Chebyshev scalarization[82] has the capacity to overcome the problem of non-convex Pareto front in certain conditions. The objective target of Chebyshev scalarization

is formulated as following.

$$f(\mathbf{x}) = \max_{1 \leqslant j \leqslant d} \omega^j (\mathbf{x}^j - z^j), \forall \mathbf{x} \in \mathscr{X}$$

where $\mathbf{z} = (z^1, \ldots, z^d)$ is a reference point that dominates all other the optimal vector $\mathbf{x}^*$. For each objective j, this reference point is the maximum of the current optimal plus a small positive value, $\epsilon^j > 0$. Then:

$$z^j = \max_{1 \leqslant i \leqslant n} \mathbf{x}^j + \epsilon^j, \forall j \tag{4.2}$$

[42] shows that all the optimal solutions of Pareto front can be found by moving the reference point z.

The optimum solution $\mathbf{x}^*$ is the variable for which the scalarization function f, linear or Chebyshev, attains its maximum value

$$f(\mathbf{x}^*) := \max_{\mathbf{x} \in \mathscr{X}} f(\mathbf{x}) \tag{4.3}$$

Ordered Weighted Averaging aggregation method (OWA) was proposed by Ronald R. Yager[105] 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 [26, 68, 76, 104, 106].

Definition 4.8. An OWA operator of dimension d is a mapping $F : \mathbb{R}^d \to \mathbb{R}$, that has an associated n vector

$$\mathbf{w} = \{w_1, \ldots, w_d\}$$

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,

$$F(\mathbf{x}) = \sum_{i=1}^{d} w_i x_{\sigma(i)} = w_1 x_{\sigma(1)} + \cdots + w_d x_{\sigma(d)}$$

where $\sigma(i)$ is according to the order of $\mathbf{x}$ all elements.

A fundamental aspect of this operator is the re-ordering step, in particular an aggregate $x_{\sigma i}$ is not associated with a fixed and particular weight w_i but rather a weight is associated with a particular ordered position. Different OWA operators are distinguished by their different weights. We her point out some special cases of OWA weights:

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

$$\mathbf{MAX}(\mathbf{x}) = \max\{x_1,\ldots,x_d\}.$$

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

$$\mathbf{MIN}(\mathbf{x}) = \min\{x_1,\ldots,x_d\}$$

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

$$F_A(\mathbf{x}) = \frac{x_1+\cdots+x_d}{d}$$

To solve the problem of MOO, OWA chooses several weights function following the reference, and optimizes the objective target:

$$F(\mathbf{x}^*) := \max_{\mathbf{x}\in\mathscr{X}} f(\mathbf{x}) = \max_{\mathbf{x}\in\mathscr{X}} \sum_{i=1}^{d} w_i x_{\sigma(i)} \tag{4.4}$$

To compare the effect of these three scalarized methods, we take some simulations with different Multi-objective sets. See Fig 4.1, Fig 4.2, Fig 4.3, each figure contains four graphs, they are in order as the original Pareto front, the linear scalarization, owa aggregation and Chebyshev. From the result, the Pareto front by Chebyshev scalarization is most approximate to the original one. And it is obviously that the linear scalarization mostly does not work on concave dataset.

4.2 Multi-Objective Optimization in Bandit environment

MOO in Section 4.1 is an optimization problem in a stable environment. If the objective space is allocated in a stochastic and uncertain environment, MOO problem becomes a particular Bandit problem, Multi-Objective Multi-Armed Bandit problem (MOMAB). To solve this problem, it should consider the objective vector space and import some useful techniques from MOO into MAB algorithms.

As MOO, A reward vector of MOMAB can be optimal in one objective and sub-optimal in others. To optimize MOMAB, it should find the Pareto front who contains all optimal arms according to their reward vectors. Refer to the Multi-Objective Optimization problem, the Pareto front of MOMAB can be identified by the dominance method or an aggregation method. Here, we construct a model of MOMAB problem.

Consider an initial set $\mathscr{A}$ with K arms, where $K \geqslant 2$. Let the vector reward space be defined as a d-dimensional vector of $[0,1]^d$. When arm i is played, a random rewarding

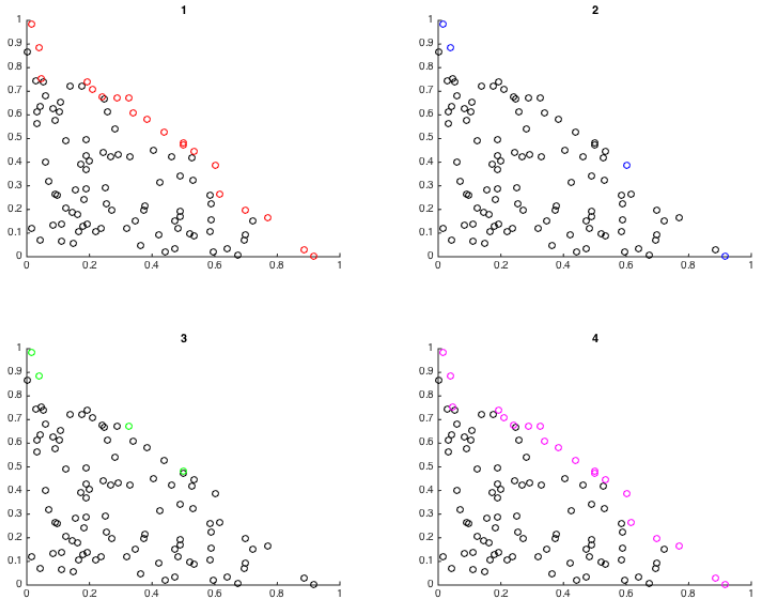

Figure 4.1: On linear datasets.

vector $\mathbf{x}$ is received, each component of this vector is drawn i.i.d from un certain distribution. For example this certain distribution is Bernoulli distribution. For arm i, its distribution is $B(\mathbf{p}_i)$. At time steps $t_1, t_2, \ldots$, the corresponding reward vectors $\mathbf{x}_{i_{t1}}^{t_1}, \mathbf{x}_{i_{t2}}^{t_2}, \ldots$ are independently and identically distributed according to the distribution with unknown expectation vector $\mathbf{p}_i = (p_i^1, \ldots, p_i^d)$. Reward values obtained from different arms are also assumed to be independent. For MOO problem, we can directly optimize the objective vector $\mathbf{p}_i$. However, the expected vector $\mathbf{p}$ is unknown to be identified. Refer to the solution of MAB, we can estimated $\mathbf{p}_i$ by an empirical expected vector $\hat{\mathbf{p}}_i$. Let $T_i(N)$ be the number that the arm i has been played during all first N plays. The expected reward vectors are computed by averaging the empirical reward vectors observed over the time. The mean reward of an arm i is estimated to $\hat{\mathbf{p}}_i(N) = \sum_{s=1}^{T_i(N)} \mathbf{x}_i(s)/T_i(N)$, where $\mathbf{x}_i(s)$ is the sampled value for arm i at time s. Here, we ideally think that if $N \to \infty$, $\mathbf{p}_i = \hat{\mathbf{p}}_i(N)$

4.2.1 Algorithms for MOMAB

In this section, we introduce some algorithms to optimize MOMAB problem. They are divided from two aspects: dominance and aggregation.

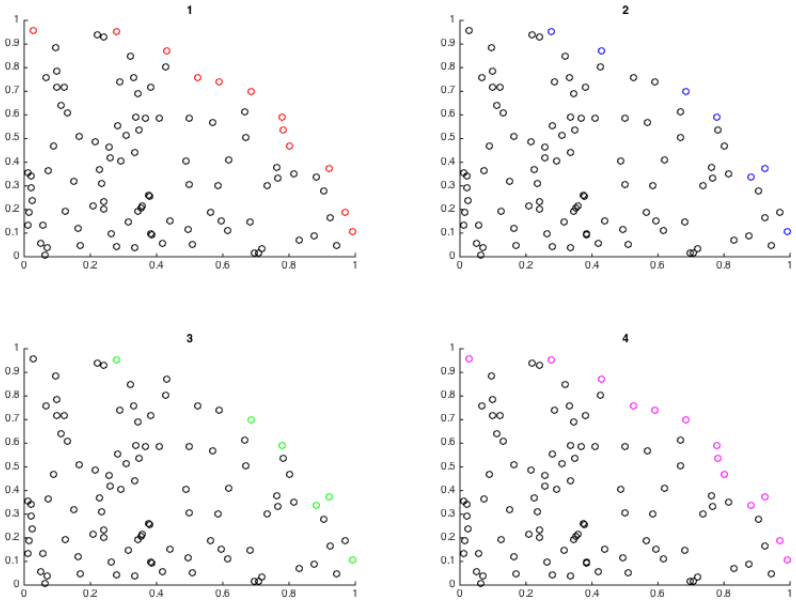

Figure 4.2: On convex datasets.

The scalarized PAC algorithm Refer to the paper[43]. This is a typical aggregation algorithm to identify the Pareto front in a stochastic environment. It tolerates an error ϵ and can find the Pareto front with probability at $1-\delta$. Here, ϵ and δ are two tolerant parameters. An arm i is optimal for a given scalarization function f_w iff

$$f_w(\hat{\mathbf{p}}_i) > \max_{k \in \mathscr{A}} f_w(\hat{\mathbf{p}}_k) - \epsilon$$

The algorithm scalarized PAC (denoted as sPAC) is given in Appendix A.13. It assumes a fixed number of weight vectors $W \leftarrow \{w_1, \ldots, w_{|W|}\}$. Under the stationary Bandit environment, the expected reward vector $\mathbf{p}_i$ of an arm i is not the same as its empirical reward vector $\hat{\mathbf{p}}_i$. There is some probability to bound the difference of $\mathbf{p}_i$ and $\hat{\mathbf{p}}_i$ with the confidence value $\epsilon > 0$ and a small error probability $\delta > 0$, where $f_w(\hat{\mu}_i) - f_w(\mu_i) > \epsilon$, for any scalarization function f_w (shown in Appendix A.13).

For the set $\mathscr{A}$, each arm of this set 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, it identifies the optimal arms using the K 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

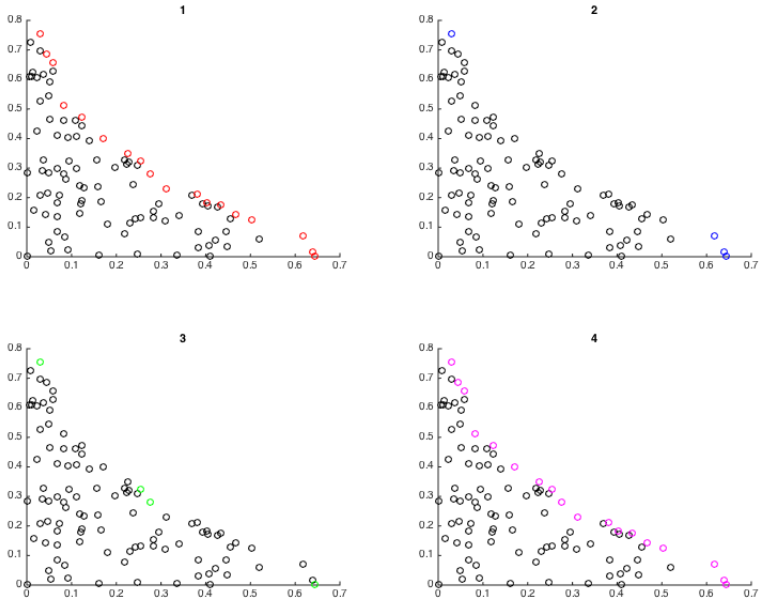

Figure 4.3: On concave datasets.

arm is added to the Pareto front $\mathscr{A}^*$. The optimal arms are maintained in Pareto front $\mathscr{A}^*$, and the dominated arms are deleted from $\mathscr{A}^*$

UCB1 in MOMAB This is a dominance method. In the MAB problems, Upper Confidence Bound(UCB) policy [11] plays firstly each arm, then adds an exploration bound to the estimated mean $\hat{\mathbf{p}}$ of each arm i. The exploration bound is an upper confidence bound which depends on the number of times arm i has been pulled. UCB pulls the optimal arm i^* that maximizes the function $\hat{\mathbf{p}}_i + \sqrt{\frac{2\ln(t)}{T_i(N)}}$ as follows:

$$i^* = \underset{1 \leqslant i \leqslant |\mathscr{A}|}{\operatorname{argmax}} \left(\hat{\mathbf{p}}_i + \sqrt{\frac{2\ln(t)}{T_i(N)}} \right)$$

where $T_i(N)$ is the number of times arm i has been pulled.

For the MOMAB problems, [42] extends 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 once. At each time step t, it estimates the mean vector of each of multi-objective arms i, i.e. $\hat{\mathbf{p}}_i = [\hat{p}_i^1, \ldots, \hat{p}_i^d]$ 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:

$$\forall j \in \{1,\dots K\}, \hat{p}_k^j + \sqrt{\frac{2\ln(t\sqrt{\lceil 4]d|\mathscr{A}^*|)}}{T_k(N)}} \not\succ \hat{p}_i^j + \sqrt{\frac{2\ln(t\sqrt{\lceil 4]d|\mathscr{A}^*|)}}{T_i(N)}}$$

Pareto-UCB uniformly and randomly selects one arm of the set $\mathscr{A}^*$. During the exploration phase, the principal idea is to select more times an arm $j \notin \mathscr{A}^*$ that is closer to the Pareto front set than the arm $k \notin \mathscr{A}^*$ that is far from $\mathscr{A}^*$. During the exploitation phase, it should pull most of times the optimal arm in the Pareto optimal set $\mathscr{A}^*$.

Annealing Linear Scalarized Based MOMAB algorithm This algorithm is proposed by [107]. It aggregates Multi-objective space 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 top 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. (shown in Appendix A.14)

Part II

Contributions

Chapter 5

Passive-Aggressive Classification with Bandit Feedback

In this chapter, we address the problem of Classification with Bandit Feedback, which has been introduced in chapter 3. Based on the Passive-Aggressive algorithm, we here propose some novel algorithms to solve the problems from Multi-class, Multi-label and Reproducing Kernel Hilbert Space. Thanks to an effective framework of Online Passive-Aggressive, our algorithms can perform max-margin for online learning. By analyzing theoretically, we find the upper bound of cumulative squared loss, some of them even reached the level as same as Online Passive-Aggressive. Finally, we provide several exprimentations to compare the novel algorithms with synthetic and real world datasets.

5.1 Multi-class PA with Bandit feedback

In Chapter 3, we have introduced some multiclass classification algorithm with Bandit feedback, i.e. Banditron, Confidit. Here, we present a novel algorithm PAB[111] 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 a random draw according to the current weight matrix w_t from the label set $[1,\ldots,K]$ (refer to Eq. 3.1). Unlike the conventional learning paradigm, if $\hat{y}_t \neq y_t$, it needs to perform an *exploration*, here we use ϵ-greedy (see in section 2.3.4) to sample $\tilde{y}_t$ instead of $\hat{y}_t$. It samples according to a probability distribution $\mathbb{P}(\tilde{Y}|\hat{y}_t)$

$$\forall i \in \{1,\ldots,K\}, \mathbb{P}(\tilde{Y}_t = i|\hat{y}_t) = \mathbb{1}(\hat{y}_t = i)\cdot(1-\epsilon) + \frac{\epsilon}{K} \tag{5.1}$$

The parameter $\epsilon \in [0,1]$ in the probability $\mathbb{P}$ is an exploration factor, i.e. every label will be sampled with same probabilities $\frac{\epsilon}{K}$ except the label $\hat{y}_t$ with $1-\epsilon+\frac{\epsilon}{K}$.

The above intuitive argument is formalized to define the update matrix. Let $\tilde{U}_t$ be the update matrix with a random prediction $\tilde{y}_t$. By recalling the section 3.1.1, the update for PA algorithm is U_{PA}, where

$$U_{PA}(x_t, \hat{y}_t, y_t) = \tau_t(\Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t))$$

with

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

and

$$\tau_t = \frac{l_t}{\| \Phi(x_t, y_t) - \Phi(x_t, \hat{y}_t) \|^2 + \frac{1}{2C}}.$$

We show in the following that the expectation of PAB update equals to the one of PA. PAB starts with the initiation of matrix $w_1 = \mathbf{0}$. Its update contains two items:

$$\begin{aligned} w_{t+1} &= w_t + U_{PAB}(x_t, \hat{y}_t, \tilde{y}_t) = w_t + U_{t,1} + U_{t,2} \\ U_{t,1} &= \frac{\mathbb{1}(\tilde{y}_t = y_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)} U_{PA}(x_t, \hat{y}_t, \tilde{y}_t) \\ U_{t,2} &= \frac{\mathbb{1}(\tilde{y}_t = y_t) - \mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)}{\mathbb{P}(\tilde{Y} = \tilde{y}_t | \hat{y}_t)} \cdot \rho_c \frac{\Phi(x_t, \hat{y}_t)}{2 \| x_t \|^2 + \frac{1}{2C}} \end{aligned} \tag{5.2}$$

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.(5.2) and let $U_{PA}(x_t, \hat{y}_t, y_t)$ be defined according to eq.(3.7). 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 has no capacity to learn on real and noisy data. So, we add an item $U_{t,2}$ to the update of Simple PAB, for increasing the stability of algorithm and reducing 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.(5.2), $\mathbb{E}_{\tilde{Y}}[U_{t,2}] = 0$.*

Proof. For each round t, we have

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

■

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

Proof.

$$\begin{aligned}
\mathbb{E}_{\tilde{Y}}[\langle U_{t,1}, U_{t,2}\rangle] &= \sum_{i=1}^{K} \mathbb{P}(i|\hat{y}_t) \left\langle \mathbb{1}(i = y_t) \frac{U_{PA}(x_t, \hat{y}_t, i)}{\mathbb{P}(i|\hat{y}_t)}, \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y})} \frac{\rho_c \Phi(x_t, \hat{y}_t)}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}} \right\rangle \\
&= \sum_{i=1}^{K} \mathbb{1}(i = y_t) \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\rho_c}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}} \langle U_{PA}(\mathbf{x}_t, \hat{y}_t, i), \Phi(\mathbf{x}_t, \hat{y}_t)\rangle \\
&= \sum_{i=\hat{y}_t} \mathbb{1}(\hat{y}_t = y_t) \frac{\mathbb{1}(\hat{y}_t = y_t) - \mathbb{P}(\hat{y}_t|\hat{y}_t)}{\mathbb{P}(\hat{y}_t|\hat{y}_t)} \frac{\rho_c}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}} \langle U_{PA}(\mathbf{x}_t, \hat{y}_t, \hat{y}_t), \Phi(\mathbf{x}_t, \hat{y}_t)\rangle \\
&+ \sum_{i \neq \hat{y}_t} \mathbb{1}(i = y_t) \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\rho_c}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}} \langle U_{PA}(\mathbf{x}_t, \hat{y}_t, i), \Phi(\mathbf{x}_t, \hat{y}_t)\rangle
\end{aligned}$$

with $\langle \Phi(\mathbf{x}_t, y), \Phi(\mathbf{x}_t, z)\rangle = \parallel \mathbf{x}_t \parallel^2$ if $y = z$, and 0 for others (refer to the definition of feature vector in section 3.1.1).

Due to

$$U_{PA}(\mathbf{x}_t, \hat{y}_t, \hat{y}_t) = \tau_t(\Phi(\mathbf{x}_t, \hat{y}_t) - \Phi(\mathbf{x}_t, \hat{y}_t)) = \mathbf{0}$$

and $\forall i \neq \hat{y}_t$,

$$\langle U_{PA}(\mathbf{x}_t, \hat{y}_t, i), \Phi(\mathbf{x}_t, \hat{y}_t)\rangle = \tau_t \langle \Phi(\mathbf{x}_t, i) - \Phi(\mathbf{x}_t, \hat{y}_t), \Phi(\mathbf{x}_t, \hat{y}_t)\rangle = -\tau_t \parallel \mathbf{x}_t \parallel^2 .$$

$$\mathbb{E}_{\tilde{Y}}[\langle U_{t,1}, U_{t,2}\rangle] = -\sum_{i \neq \hat{y}_t} \mathbb{1}(i = y_t) \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\rho_c \parallel \mathbf{x}_t \parallel^2}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}}$$

where the following items s_i are always positive.

$$s_i = \mathbb{1}(i = y_t) \frac{\mathbb{1}(i = y_t) - \mathbb{P}(i|\hat{y}_t)}{\mathbb{P}(i|\hat{y}_t)} \frac{\rho_c \parallel \mathbf{x}_t \parallel^2}{2 \parallel \mathbf{x}_t \parallel^2 + \frac{1}{2C}}$$

$$\mathbb{E}_{\tilde{Y}}[< U_t^1, U_t^2 >] = -\sum_{i \neq \hat{y}_t} s_i \leq 0$$

■

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* $t = 1, ..., T$ ***do***

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

Set $\hat{y}_t = \underset{r\in\{1,\dots,K\}}{argmax}\langle w_t, \Phi(x_t, r)\rangle$

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

Randomly sample $\tilde{y}_t$ *according to* $\mathbb{P}(\tilde{Y}_t = i|\hat{y}_t)$
Receive the feedback $\mathbb{1}(\tilde{y}_t = y_t)$

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

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

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

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 [62]. 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 [73]. This set is made of 47236-dimensional vectors, contains 4 classes (to choose the first level categorization.) with the size of 10^5. To different datasets, the parameters of different algorithms refer to Table 5.1.

In the Figure 5.1 gives the cumulative errors 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).

Table 5.1: The summary of algorithm parameters for different datasets. B. denotes Algorithm Banditron, C. is Confidit, PG presents Policy Gradient and S.PAB is Simple PAB.

Dataset	**B.**	**C.** ($\alpha = 1$)	**PG** ($\lambda = 10^{-3}$)	**S.PAB** ($\rho = 0$)	**PAB** ($\rho = 1$)
SynSep	$\epsilon = 0.014$	$\eta = 10^3$	$\eta = 0.01$	$C = 10^{-3}, \epsilon = 0.7$	$C = 10^{-3}, \epsilon = 0.7$
SynNonSep	$\epsilon = 0.006$	$\eta = 10^3$	$\eta = 0.01$	$C = 10^{-5}, \epsilon = 0.7$	$C = 10^{-5}, \epsilon = 0.7$
Reuters	$\epsilon = 0.05$	$\eta = 10^2$	$\eta = 0.1$	$C = 10^{-4}, \epsilon = 0.6$	$C = 10^{-4}, \epsilon = 0.6$

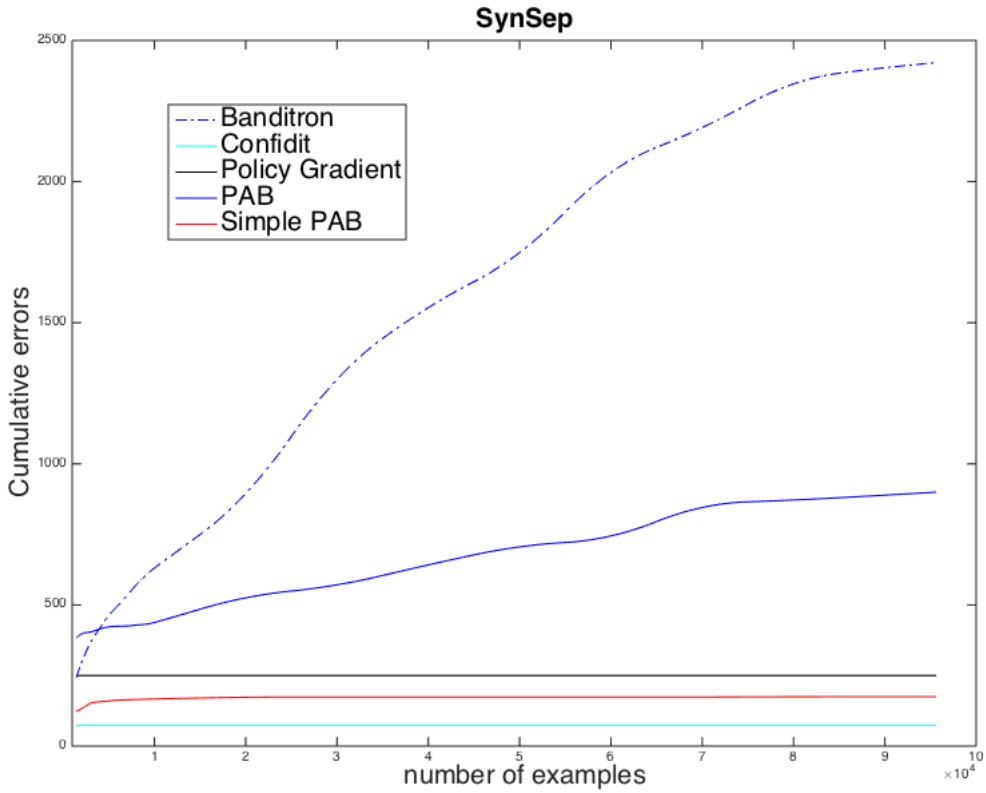

Figure 5.1: Cumulative errors of each algorithm under the SynSep data.

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 [36] 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

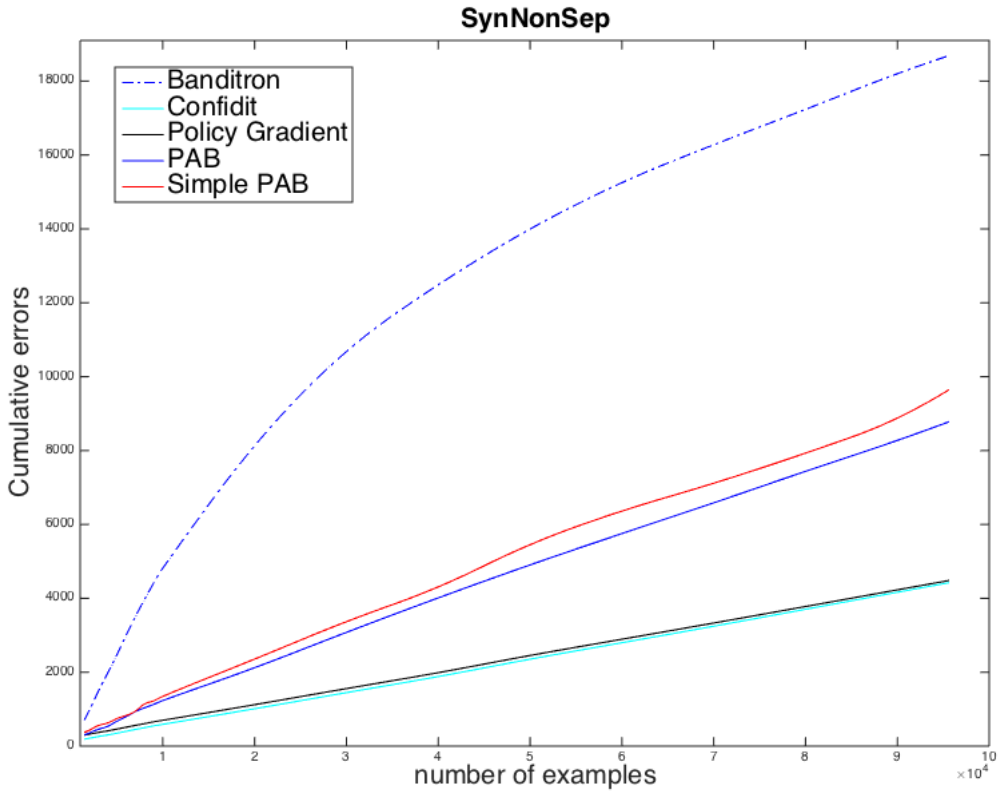

Figure 5.2: Cumulative errors of each algorithm under the SynNonSep data.

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.

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

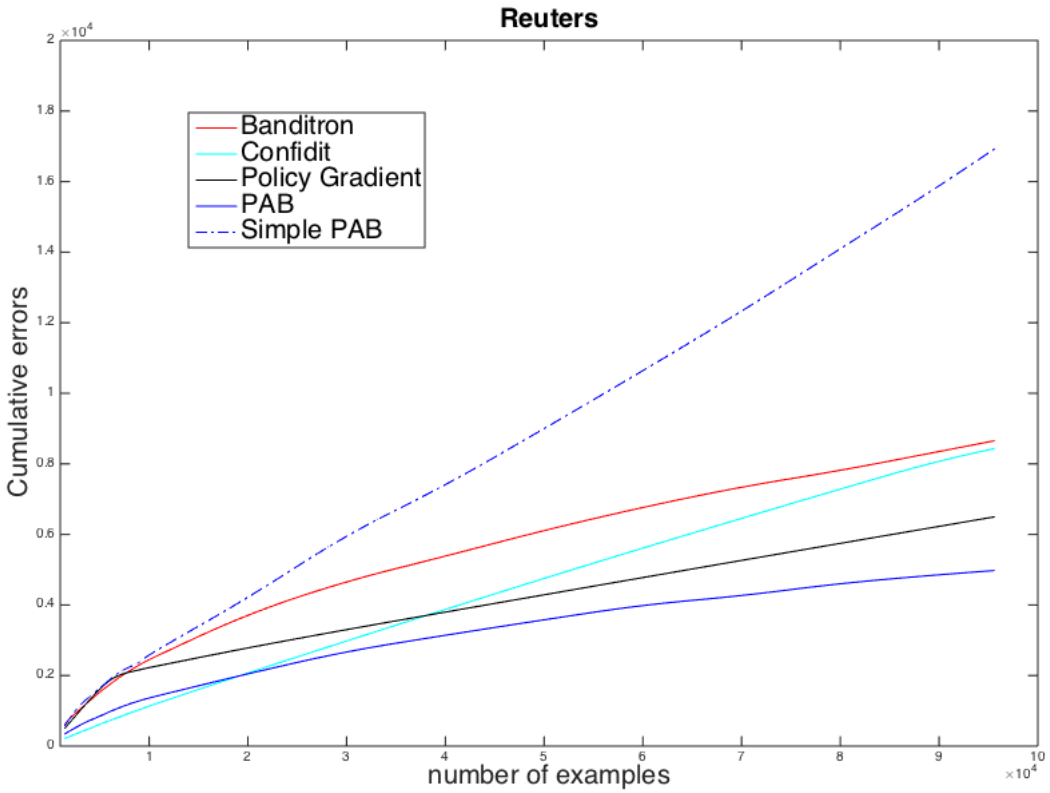

Figure 5.3: Cumulative errors of each algorithm under the RCV1-v2 data.

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 ϵ, 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

Different to the PAB algorithm, BPA [110] also an adaptation of PA, its error even 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 score of $\langle w_t, \Phi(x_t, i)\rangle$, is defined as below:

$$\hat{y}_t = h_t(x_t) = \underset{i\in\{1,\ldots,K\}}{\operatorname{argmax}} \langle w_t, \Phi(x_t, i)\rangle \tag{5.3}$$

where $w_t \in \mathbb{R}^{K\times d}$ like previously. Like previously, it needs to perform an exploration, i.e., sample a label randomly $\{1,\ldots,K\}$ with parameter ϵ 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 (see eq. 5.1) The instantaneous loss is given by the following function,

$$l_t = [\gamma + \left(1 - 2\mathbb{1}_{\tilde{y}_t=y_t}\right) \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+ \tag{5.4}$$

Here, we define the parameter γ equals to 1. Otherwise $\left(1-2\mathbb{1}_{\tilde{y}_t=y_t}\right)$ 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 $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \geqslant 1$ and then increases for decreasing values of $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$. In contrast, when the prediction is incorrect, the loss is equal to $[1 + \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$, i.e., stays at 0 for $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \leq -1$ and then increases for increasing values of $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$.

The linear classifiers are updated at each trial using the standard tools from convex analysis[22], Where w_t satisfies the constraint in Eq. 3.5.

$$L(w, \tau) = \frac{1}{2} \parallel w - w_t \parallel^2 + \tau \left(1 + (1 - 2\mathbb{1}_{\tilde{y}_t=y_t}) \langle w_t, \Phi(x_t, i)\rangle\right) \tag{5.5}$$

$$w_{t+1} = w_t + \tau \left(2\mathbb{1}_{\tilde{y}_t=y_t} - 1\right) \Phi(x_t, \tilde{y}_t) \tag{5.6}$$

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 [99] and a non-negative slack variable ξ is introduced into the optimization problem in Equation 3.5. According with [33], 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:

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

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

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

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

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

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

Set $\hat{y}_t = \underset{i=1,...,K}{argmax} \langle w_t, \Phi(x_t, i) \rangle$

for all $i \in [1, ..., K]$ ***do***

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

end for

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

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

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

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

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* $(x_1, y_1), ..., (x_T, y_T)$ *be a sequence of separable examples where* $x_t \in \mathbb{R}^d$, $y_t \in \mathscr{Y}$ *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,*

$$\sum_{t=1}^{T} l_t^2 \leqslant R^2 \cdot \| u \|^2 \tag{5.7}$$

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 $w_1 = \vec{0}$,

$$\sum_{t=1}^{T} \Delta_t = \| u \|^2 - \| w_{t+1} - u \|^2 \leqslant \| u \|^2 \tag{5.8}$$

Using the definition of update :

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

With $l_t = [1 + (1 - 2\mathbb{1}_{(\tilde{y}_t = y_t)}) \cdot \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$ and $l_t^* = [1 + (1 - 2\mathbb{1}_{(\tilde{y}_t = y_t)}) \cdot \langle u, \Phi(x_t, \tilde{y}_t)\rangle]_+$, 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 $\forall t \in [1, ..., T]$, $l_t^* = 0$,

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

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

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

■

Theorem 5.2. *Let $(x_1, y_1), ..., (x_T, y_T)$ be a sequence of non-separable examples where $x_t \in \mathbb{R}^d$, $y_t \in [1, ..., K]$ and $\| x_t \| \leqslant R$ for all t. Then for any matrix $u \in \mathbb{R}^{K \times d}$ 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, we denote $L_t = \sqrt{\sum_{t=1}^{T} l_t^2}$ and $U_t = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}$,

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

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

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

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

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

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

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

■

5.2.2 Experiments

Here, we evaluate the algorithms over two synthetic and three real world data sets. Their characteristics are summarized in Table 5.2.

Table 5.2: Summary of the five data sets, including the numbers of instances, features, labels and whether the number of examples in each class are balanced.

Dataset	**Instances**	**Features**	**Labels**	**Balanced**
SynSep	10^5	400	9	Y
SynNonSep	10^5	400	9	Y
RCV1-v2	10^5	47236	53	N
Letter	$2 * 10^4$	16	26	N
Pen-Based	$1.32 * 10^4$	16	10	N

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 [62]. 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[73]. The original data set is composed by multi-label instances. So we make some preprocessing likes [18].

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 [5, 60]. 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 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 $\times$ 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. 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 supervised algorithms. To different datasets, the parameters of different algorithms refer to Table 5.3.

Table 5.3: The summary of algorithm parameters for different datasets. P. denotes Perceptron, PA is Passive-Aggressive online algorithm, B. is Banditron, C. is Confidit and BPA.

Dataset	**P.**	**PA**	**B.**	**C.**	**BPA**
SynSep	null	$C = 0$	$\epsilon = 0.014$	$\eta = 10^3$	$\epsilon = 0.4, C = 0$
SynNonSep	null	$C = 10^{-2}$	$\epsilon = 0.65$	$\eta = 10^3$	$\epsilon = 0.8, C = 10^{-2}$
Reuters	null	$C = 10^{-2}$	$\epsilon = 0.4$	$\eta = 10^2$	$\epsilon = 0.2, C = 10^{-2}$
LR(26 letters)	null	$C = 0.1$	$\epsilon = 0.2$	$\eta = 10^2$	$\epsilon = 0.8, C = 1$
LR(10 numbers)	null	$C = 0.1$	$\epsilon = 0.4$	$\eta = 10$	$\epsilon = 0.6, C = 1$

Figures 5.6, 5.7 and 5.8 present the result on three real data sets. With the three real data sets, the supervised algorithms, 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 ϵ 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 ϵ on the different data sets. In contrast with Banditron, BPA shows that ϵ has a very little influence on the final error rate, indicating a capability to deal with small exploration rates.

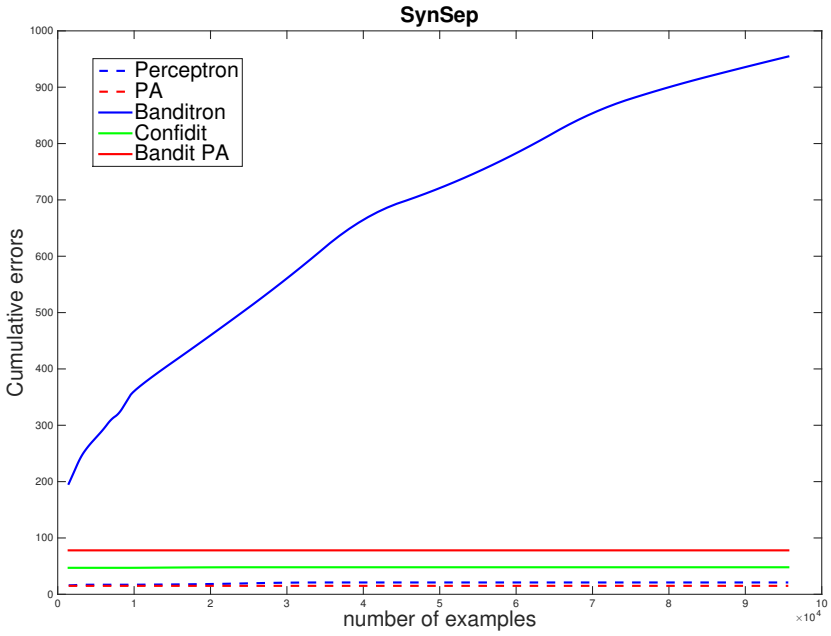

Figure 5.4: Cumulative Errors on the synthetic data set of SynSep.

5.2.3 Conclusion

In this section, we proposed a novel algorithm for online multiclass classification 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

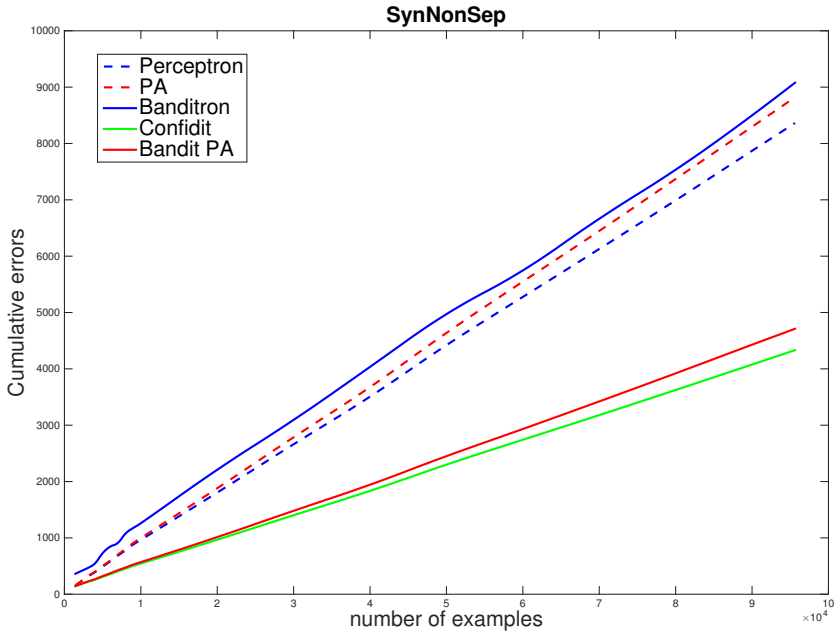

Figure 5.5: Cumulative Errors on the synthetic data set of SynNonSep.

sets non-separable. In the next section, we will take BPA to deal with non-linear data sets by combining the Kernel method.

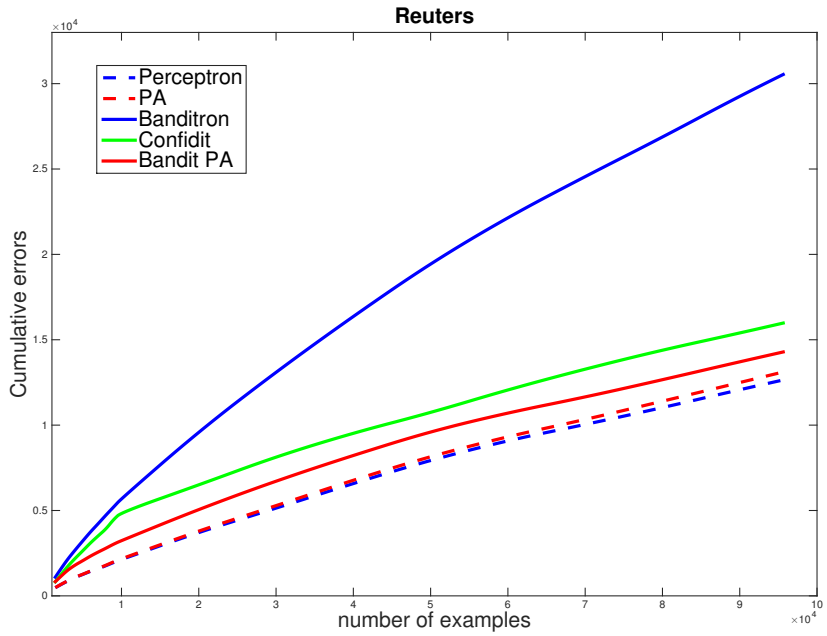

Figure 5.6: Cumulative Errors on the real data set of RCV1-v2 (53 classes).

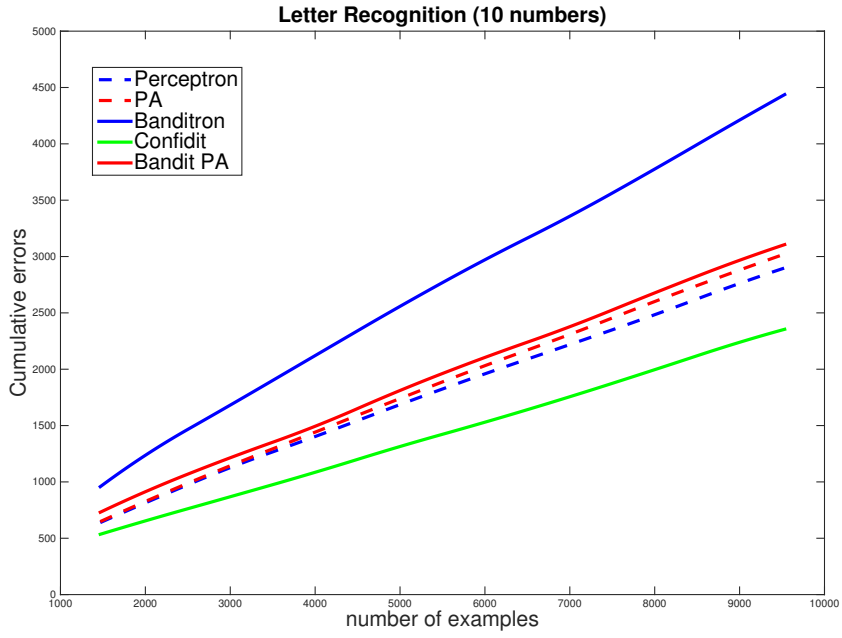

Figure 5.7: Cumulative Errors on the real data set of Letter Recognition (10 numbers).

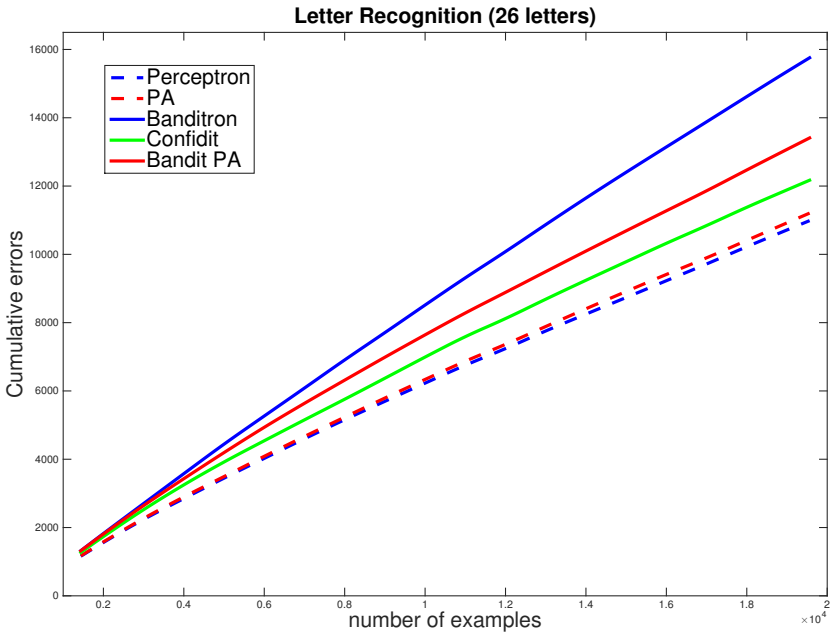

Figure 5.8: Cumulative Errors on the real data set of Letter Recognition (26 Letters).

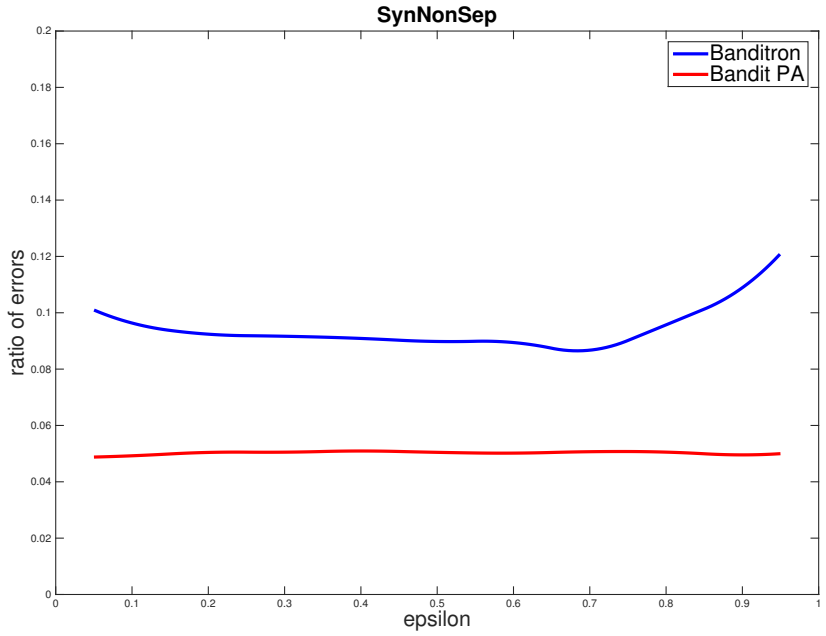

Figure 5.9: Average error of Banditron and BPA for parameter's value ϵ on the data set of SynNonSep.

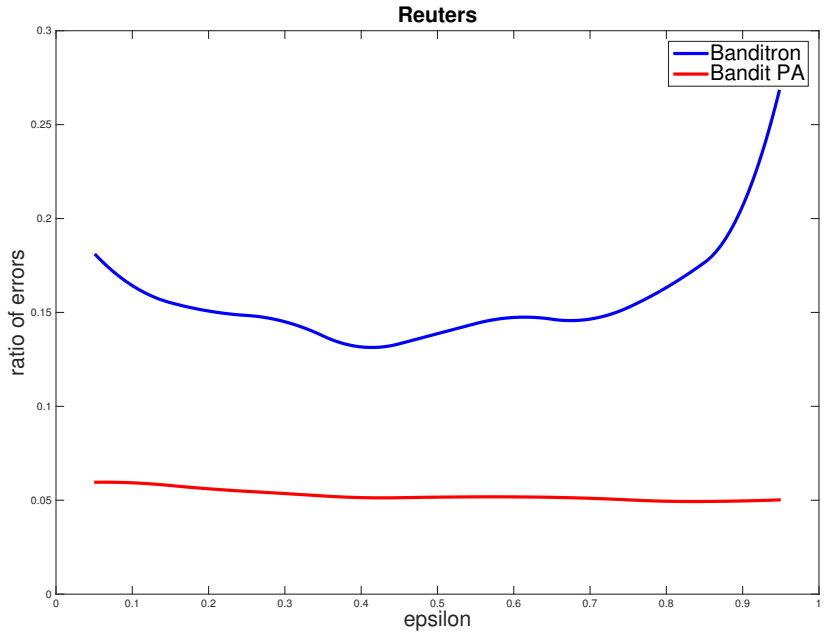

Figure 5.10: Average error of Banditron and BPA for parameter's value ϵ on the data set of Reuters.

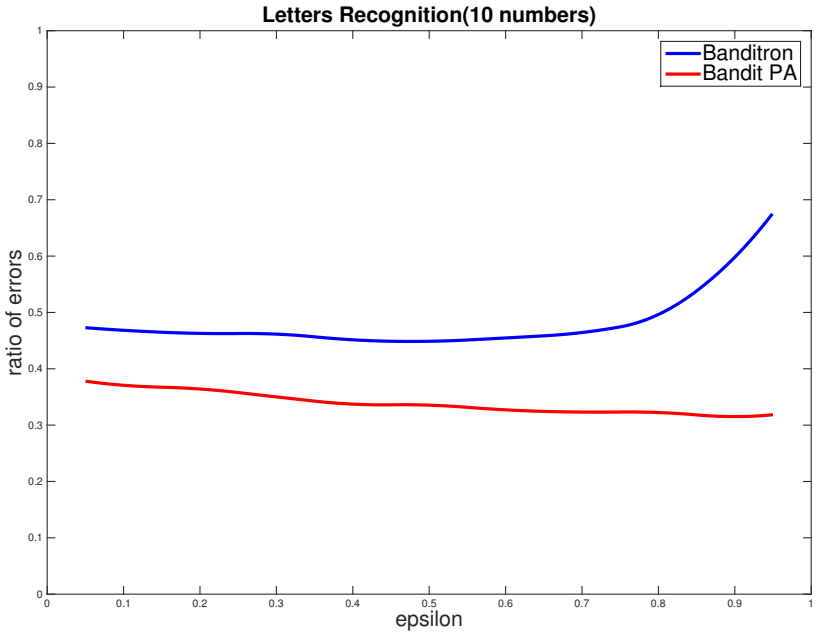

Figure 5.11: Average error of Banditron and BPA for parameter's value ϵ on the data set of Letter Recognition.

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 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 and more researcher. In [64, 88, 92], 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 learning in a Reproducing Kernel Hilbert Space, by considering the classical stochastic gradient descent.

In this section, we proposed two kernel based algorithms 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:

$$\hat{y}_t = \underset{i \in [1,..,K]}{\arg\max} f^i(x).$$

Considering the update of BPA algorithm Function 5.6, 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

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

We need take attention to this formulate of loss function, it is different to the loss function in Section 5.1 and Section 5.2. It takes an indicator instead of a constant, which also works

to maximize the margin. Our goal is to reduce the computational complexity and avoid some useless support vector.

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

$$\forall k \in \{1,\dots,K\}, f_t^k = \sum_{i=1}^{t-1} \mathbb{1}(k=\tilde{y}_i)\tau_i \cdot (2\mathbb{1}(\tilde{y}_i = y_i) - 1) \cdot k(x_i, \cdot) \tag{5.9}$$

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

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

Creat K containers $\mathscr{C} = \{C_1,\dots,C_K\}$, $\forall i \in [1,\dots,K], C_i = \mathbf{0} \in \mathbb{R}^d$

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

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

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

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

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

end for

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

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

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

if $l_t \neq 0$ ***then***

$C_{\tilde{y}_t} \leftarrow C_{\tilde{y}_t} \cup \{\mathbf{x}_t\}$ *with* $\alpha_{|\mathscr{C}+1|}^{\tilde{y}_t} = \tau_t \cdot (2\mathbb{1}(\tilde{y}_t = y_t) - 1)$

end if

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 Kernel Stochastic Gradient Descent with BPA loss

This method should refer to Kivinen's [64] 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 5.4 the loss function of algorithm BPA. Consider the classical stochastic gradient descent, take the gradient gradient to each hypothesis f^i of $\mathscr{F}$.

$$\forall k \in [K], f_{t+1}^k = f_t^k - \eta_t \partial_{f^k} R[\mathscr{F}]|_{f^k = f_t^k} \tag{5.10}$$

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,

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

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

$$f_{t+1}^k = \begin{cases} f_t^k \cdot (1 - \lambda\eta) + \eta \cdot (2\mathbb{1}_{(\tilde{y}_t = y_t)} - 1) \cdot k(x_i, \cdot) & \text{if } k = \tilde{y}_t \\ f_t^k & \text{else} \end{cases} \tag{5.11}$$

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

$$\sigma_t^k = \sum_{s=1}^{t} \mathbb{1}(\tilde{y}_s = k) \tag{5.12}$$

By the parameters σ, the update function 5.11 could be expressed as the following equation: for $\forall k \in \{1, \ldots, K\}$

$$f_{t+1}^k = \sum_{i=1}^{t} \eta \alpha_i^k \cdot k(x_i, \cdot) \tag{5.13}$$

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 5.13 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 [64] 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 $\parallel k(x, \cdot) \parallel \leqslant X$ where $\parallel \cdot \parallel$ denotes $\parallel \cdot \parallel_{\mathscr{H}}$. 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 $k \in [K]$

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

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 $x_1, \ldots, x_T$

Initialize the parameters $\lambda > 0$*, a truncation parameter* $\tau \in \mathbb{N}$*, a learning rate* $\eta \in (0, 1/\lambda)$

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

Receive the instance data x_t

Predict $\hat{y}_t = \underset{i \in \{1,\ldots,K\}}{argmax}\ f_t^i(x_t)$

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* $p_t = (p_{1,t}, \ldots, p_{K,t})$

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 Experiments

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 [6, 7]. 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'[74]. 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 algorithms Banditron (in RKHS), KBPA and KSGD to compare. In order to perform the effect of RKHS, we choose KBPA in linear model as the reference object and choose **Laplace** for the kernel function. Its form looks like the

following formulate.

$$K_{Laplace}(x,y) = \exp\left(-\frac{\| x-y \|}{\sigma}\right)$$

So, all participant algorithms contains: KBanditron, KBPA (linear), KBPA (Laplace), and KSGD (Laplace). 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 We mainly analyze these experiments from the following aspects.

Average training time for each instance: we observe the training time of every instance $\{t_1, t_2, \ldots, t_n\}$; then divide 100 ordering examples into one group $g_1 = \{t_1, \ldots, t_{100}\}$, ..., $g_i = \{t_{1+100*(i-1)}, \ldots, t_{100*i}\}$; finally, the average training time for instances of group g_i can be calculated by $\overline{t_i} = \frac{1}{100}\sum_{s=1+100*(i-1)}^{100*i} t_s$.

Average error rate: $e_i = \sum_{s=1+100*(i-1)}^{100*i} \mathbb{1}[\hat{y}_t = y_t]/100$ this measure is calculated by the same way.

Cumulative Errors: calculate the total number of past errors.

In Figure 5.12, it gives the result of average training time on based dataset "Pendigits". From this result, the training time of three kernel algorithms increases linearly along with the number of training instances. Only the linear model is stable. From the theoretical perspective, Banditron always adds a new example passively for its support vector. Algorithm KSGD only adds a new example for its support vector if its classifier makes a bad prediction, otherwise the number of support vector is limited by the truncated parameter. Algorithm KBPA adds a new example for its support vector if and only if its predicted loss not equals to zero. So its number of support vector will increase all the time until it can make good prediction with no loss.

In Figure 5.13 and Figure 5.14, accumulative errors of algorithm KBPA firstly tend to a stable, others still increase linearly. That is because KBPA accumulates all good support vectors, KSGD only accumulates several recent support vectors and Kernel Banditron always accumulates new instance as negative support vector.

In Figure 5.15, it is about the average training time on dataset "Segment". The training time of Kernel Banditron still increases linearly, while the training time of KSGD and KBPA are as stable as linear model after a small period of increasing linearly. KSGD reaches the limited number of support vector, and KBPA quickly gets enough support vectors to make a good prediction. It could show that this dataset is separable.

In Figure 5.16 and Figure 5.17, we can observe that KBPA and KSGD performed obviously better than the other two. Two kernel algorithms have ability to solve non-linear

classification with Bandit Feedback. Considering the scale of classifier, we can use more efficient algorithm KBPA if dataset is separable, otherwise we use KSGD.

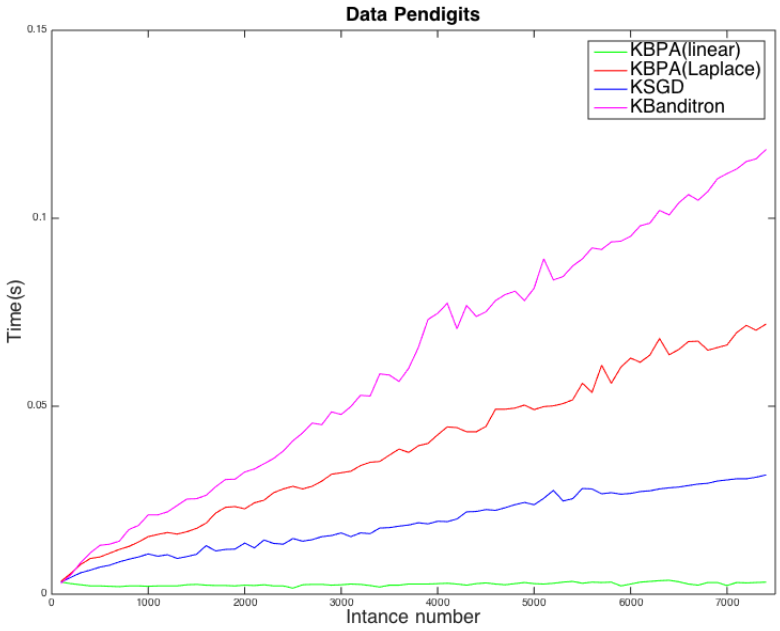

Figure 5.12: Average training time for each instance of Data Pendigits.

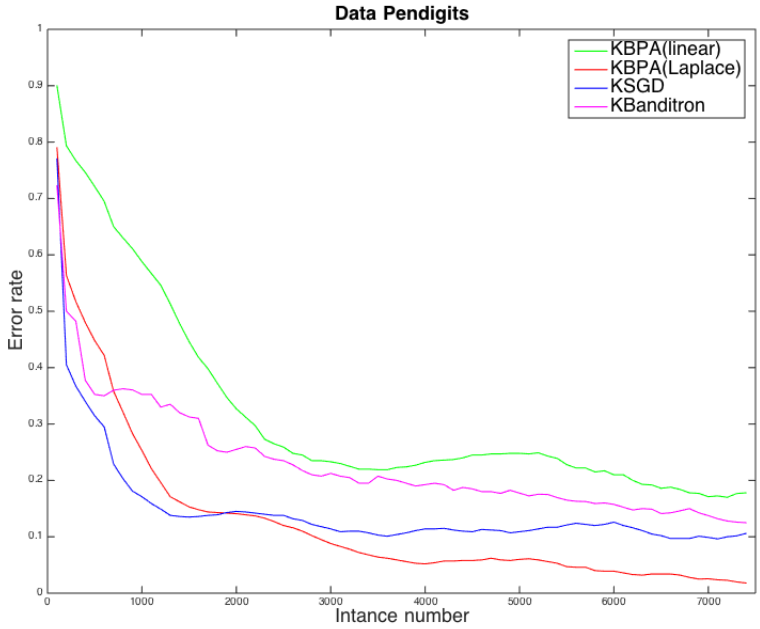

Figure 5.13: Average error rate for each instance of Data Pendigits

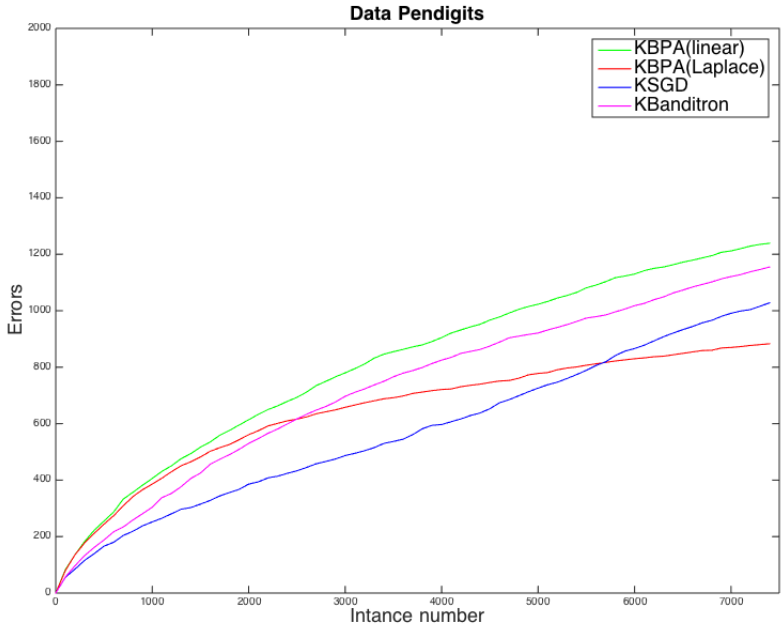

Figure 5.14: Cumulative Errors of Data Pendigits

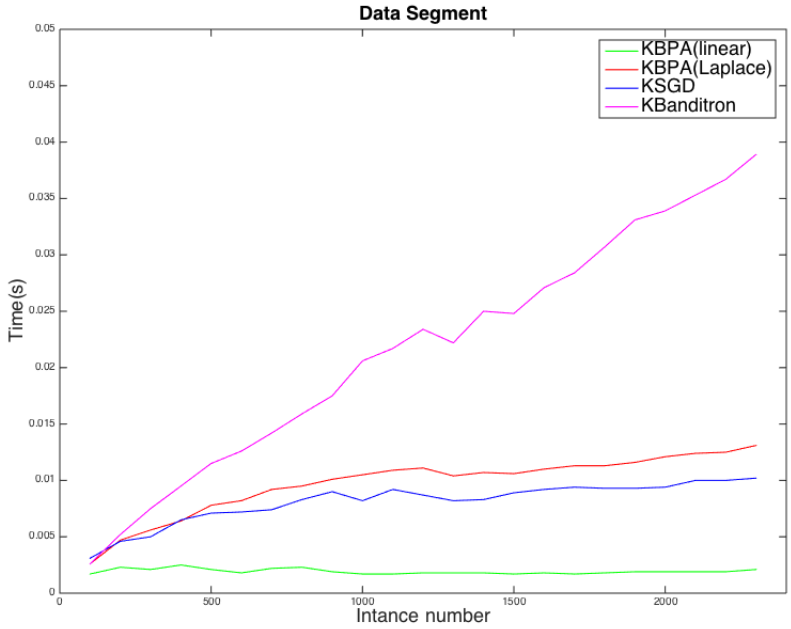

Figure 5.15: Average training time for each instance of Data Segment.

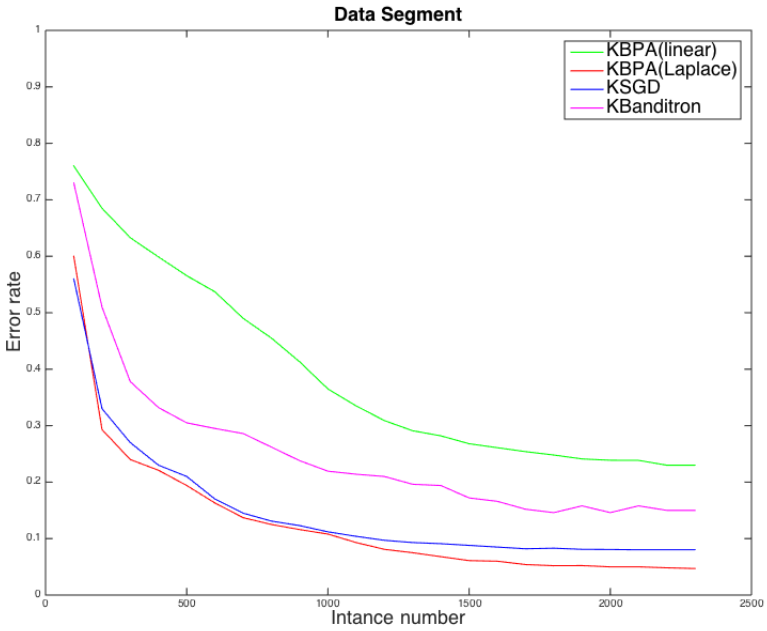

Figure 5.16: Average error rate for each instance of Data Segment

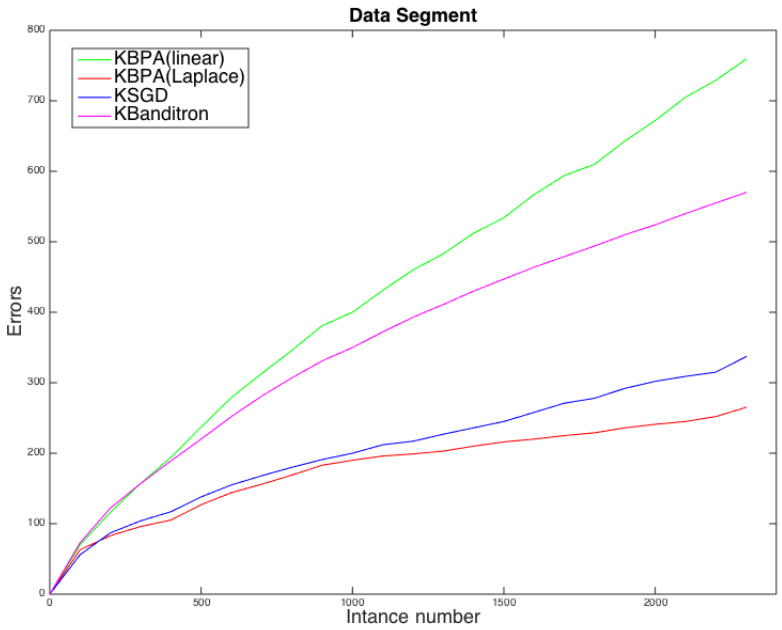

Figure 5.17: Cumulative Errors of Data Segment

5.4 Bandit PA algorithm for Multi-label Classification

5.4.1 Preliminaries

In this section, we mainly introduce the problem of Multi-label classification with Bandit feedback. In Chapter 3, we have introduced some methods. We here propose a novel algorithm based on Online Passive-Aggressive.

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

$$\beta_t^k = \begin{cases} 1 & \hat{y}_t^k = y_t^k = 1 \\ 0 & \hat{y}_t^k = 1 \& y_t^k = 0 \\ \frac{1}{2} & \hat{y}_t^k = 0 \end{cases} \tag{5.14}$$

The widely known Binary relevance Method(BM) [85] transfers the multi-label task into several single independent binary classification tasks for each label. BM approach is theoretically simple 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 compared to other methods.

We consider a multilabel classification model $f(x)$ is a mapping function $\mathscr{X} \to \{0,1\}^K$. The output of the model is a binary vector:

$$h(x) = (h_1(x), \dots, h_K(x))$$

here $h(x)$ is taken from a class of hypothesis $\mathbf{H}$ parameterized by a $K \times d$ matrix of real weights w. Hence, $\forall i \in [1,..,K], h_i(x) = \frac{1}{2}(1 + \text{sign} < w^i, \mathbf{x} >)$ 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 [33]'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. Here we use, like in previous sections, the ϵ-greedy strategy. 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:

$$p(\tilde{y}_t^i = 1|\hat{y}_t^i) = (1-\epsilon) \cdot \mathbb{1}_{(\hat{y}_t^i=1)} + \epsilon \cdot \frac{\sum_{k=1}^{K} \mathbb{1}_{(\hat{y}_t^k=1)}}{K} \tag{5.15}$$

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.

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

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

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

■

5.4.2 Analysis

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

$$l_t^k = [\mathbb{1}_{\tilde{y}_t^k=1} + \left(1 - 2\beta_t^k\right) \langle w_t, \Phi(x_t, k)\rangle]_+ \tag{5.16}$$

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 $[1 - \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$ when the prediction is positive correct: it stays at 0 for $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \geqslant 1$ and then increases for decreasing values of $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$. In contrast, when the prediction is positive incorrect, the loss is equal to $[1 + \langle w_t, \Phi(x_t, \tilde{y}_t)\rangle]_+$, i.e., stays at 0 for $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle \leq 1$ and then increases for increasing values of $\langle w_t, \Phi(x_t, \tilde{y}_t)\rangle$. 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 $l(w_t;(x_t,y_t))$ can be the hinge loss. The solution will be derived from the optimization problem:

$$\begin{aligned} w_{t+1} &= \underset{w\in\mathbb{R}^{K\times d}}{\operatorname{argmin}} \frac{1}{2} \parallel w - w_t \parallel^2 \\ s.t.\ L_t &= \sum_{k=1}^{K} l_t\left(w;(x_t,y_t^k=1)\right) = 0 \end{aligned} \tag{5.17}$$

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:

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

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.(5.15)).

The linear classifiers are updated at each trial using the standard tools from convex analysis [22]. According the instantaneous loss (see Eq.(5.16)), the constrained optimization problem of Eq.(5.17) is solved by the Lagrangian method, i.e.:

$$\mathscr{L}(w,\tau_t) = \frac{1}{2} \parallel w - w_t \parallel^2 + \tau_t \sum_{k=1}^{K} l_t^k \tag{5.18}$$

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

$$\Rightarrow w = w_t + \tau_t \sum_{k=1}^{K} \left(2\beta_t^k - 1\right)\Phi(x_t,k) \tag{5.19}$$

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

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

$$\tau_t = \frac{L_t}{\sum_{k=1}^{K} (2\beta_t^k - 1)^2 \parallel x_t \parallel^2} \tag{5.20}$$

With $L_t = \sum_{k=1}^{K} l_t^k$.

Consider for instance the common phenomenon of label noise, a mislabeled instance may cause classifiers to be drastically changed in a wrong direction like Passive-Aggressive algorithm. Refer to [99], a non-negative slack variable ξ is used to derive soft-margin into the optimization problem in Eq. 5.18.

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 $x_t \in \mathbb{R}^d$*;*
  for *each* $k = 1,2,\ldots,K$ ***do***
    Set $\hat{y}_t^k = \frac{1}{2}(1 + sign\,\langle w_t, \Phi(x_t,k)\rangle)$
  end for
  for *each* $k = 1,2,\ldots,K$ ***do***
    $\mathbb{P}(\tilde{y}_t^k = 1|\hat{y}_t^k) = (1-\epsilon)\cdot \mathbb{1}_{(\tilde{y}_t^k=1)} + \epsilon \cdot \frac{\sum_{k=1}^K \mathbb{1}_{(\hat{y}_t^k=1)}}{K}$
  end for
  Draw $\tilde{y}_t$ *randomly from* $\mathbb{P}$
  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* $\| x_t \| \leqslant R$ *for all t, and* $u \in \mathbb{R}^{K\times d}$*. Then, the cumulative squared loss of this algorithm is bounded by,*

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

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.(5.19) and (5.20),

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

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

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

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

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

■

Proof. By the proof of Theorem 5.2,

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

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

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

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

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

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

■

5.4.3 Experiments

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

Reuters RCV1-v2 collection [73], 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[47], 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 multi-class/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.

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

- **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 5.4 for the dataset RCV1-v2 and in Table 5.5 for the dataset Yeast.

Fig. 5.18, Fig. 5.19 and Fig. 5.20 show the result on the dataset RCV1-v2. We observe the algorithm OD-UCB and BPAs have better performance than PA in precision metric; however, both of two bandit algorithms play inferior to PA in recall metric. From the number of average cardinality, we can get the reason. Less cardinality has little chance to explore other labels. From the result of OneError measure, all of them optimize along with training. By Fig. 5.21, it obviously shows that the cumulative loss tends to $\sqrt{T}$ as theoretical result.

Fig. 5.22, Fig. 5.23 and Fig. 5.24 show the result on the dataset Yeast. After a phase of training, OneError measure of BPAs declines more quickly than others. So it has the most effective optimization.

By the advantage of PA max-margin principle, BPAs appears effective to address the bandit setting for multi-label classification. Its main advantage is its linear complexity in space allows to deal with high dimensional data sets and a large number of labels, on the contrary to 2nd-order descend method. The practicability of this algorithm is verified theoretically by showing competitive loss bound. Moreover, Experimental evaluation shows that BPAs performs very well, it is flexibility to control the performance between the precision and the recall by changing ϵ value. And in some dataset, it is even better than the algorithm based on 2nd-order descend. For the OneError metric, it could converge on the bound of the algorithm PA working in full-side information.

Table 5.4: 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	$2.13 * 10^{-1}$

Table 5.5: 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	$3.88 * 10^{-5}$

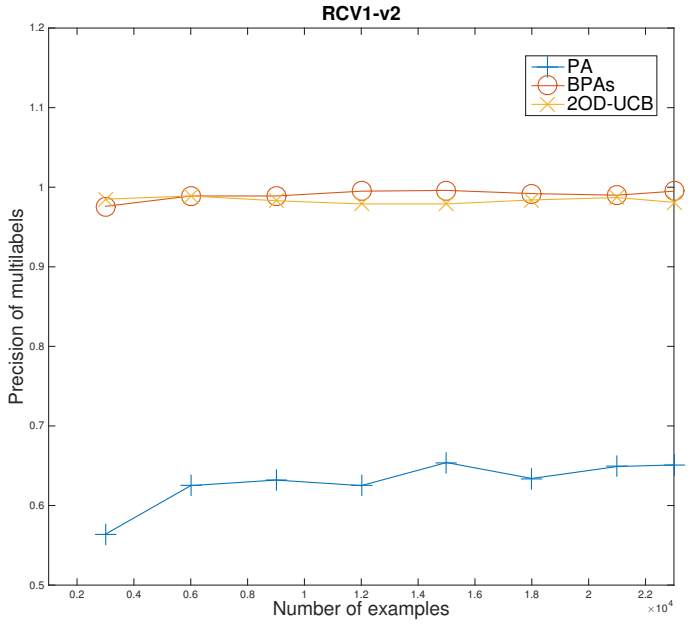

Figure 5.18: Precision of algorithms on RCV1-v2

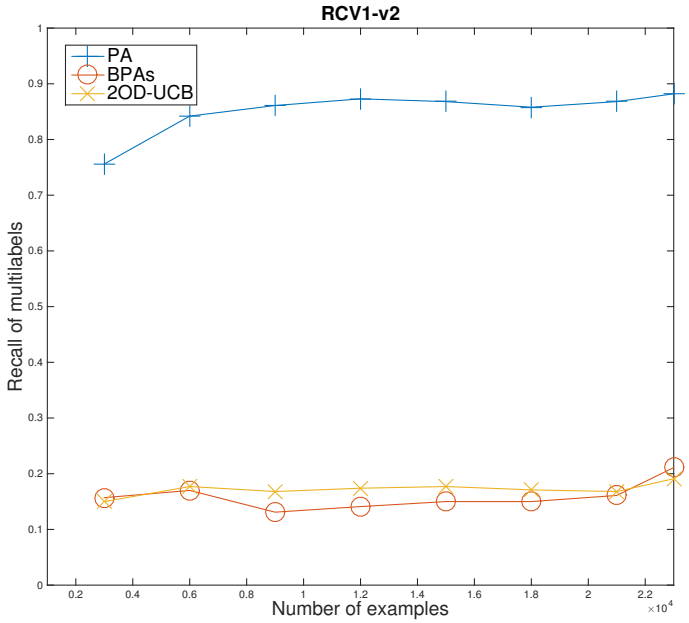

Figure 5.19: Recall of algorithms on RCV1-v2

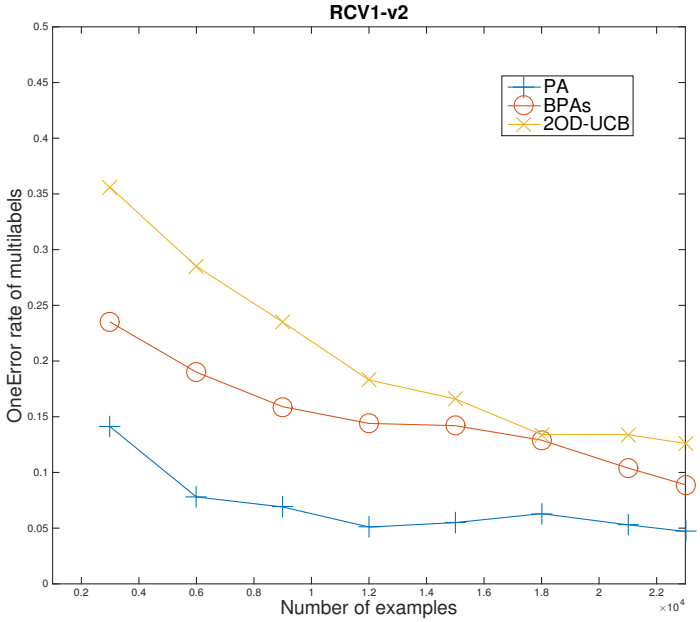

Figure 5.20: OneError of algorithms on RCV1-v2

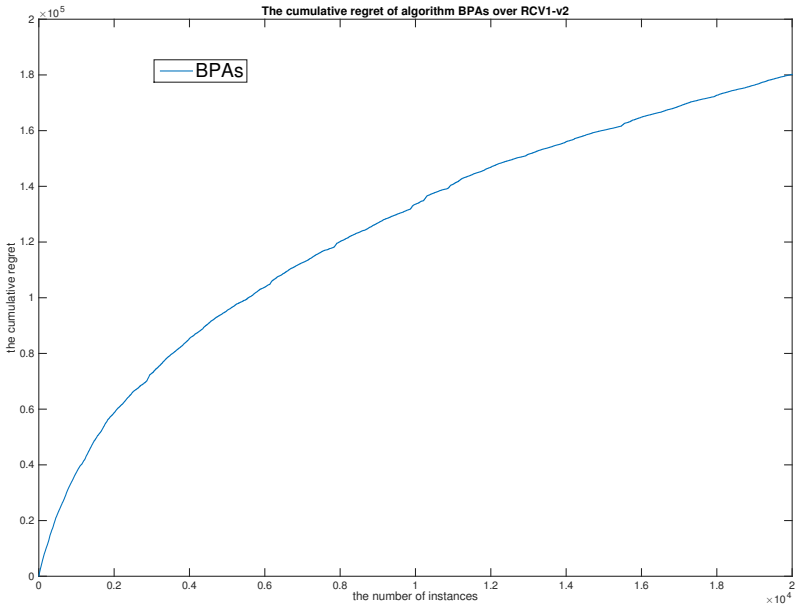

Figure 5.21: The cumulative loss of algorithms on RCV1-v2

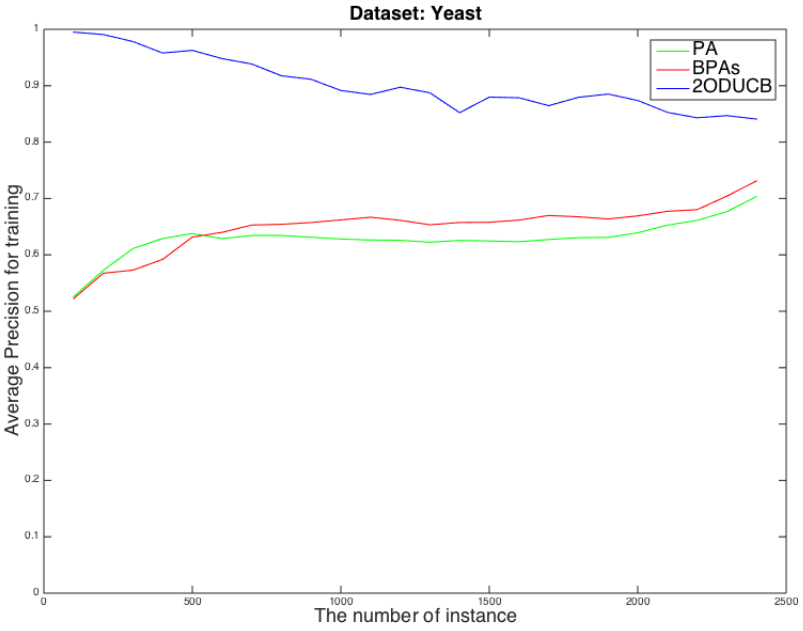

Figure 5.22: Precision of algorithms on Yeast

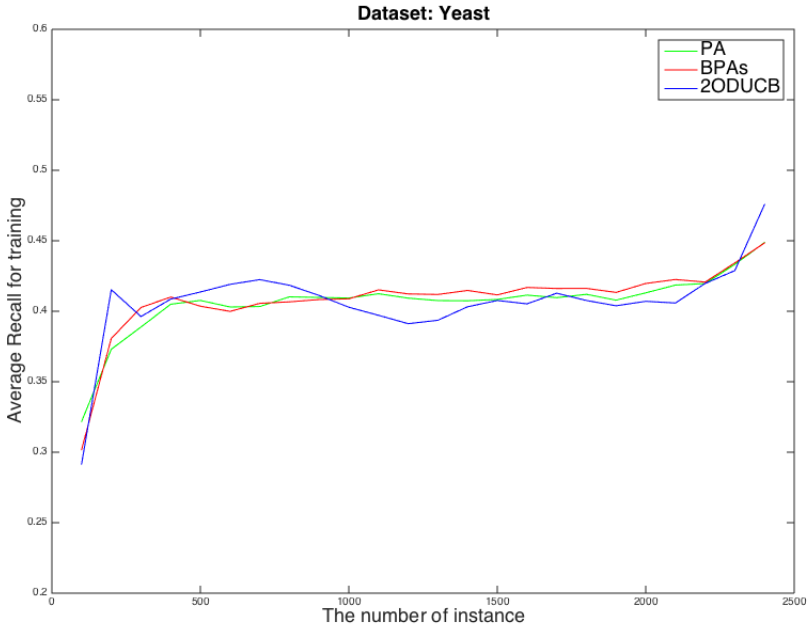

Figure 5.23: Recall of algorithms on Yeast

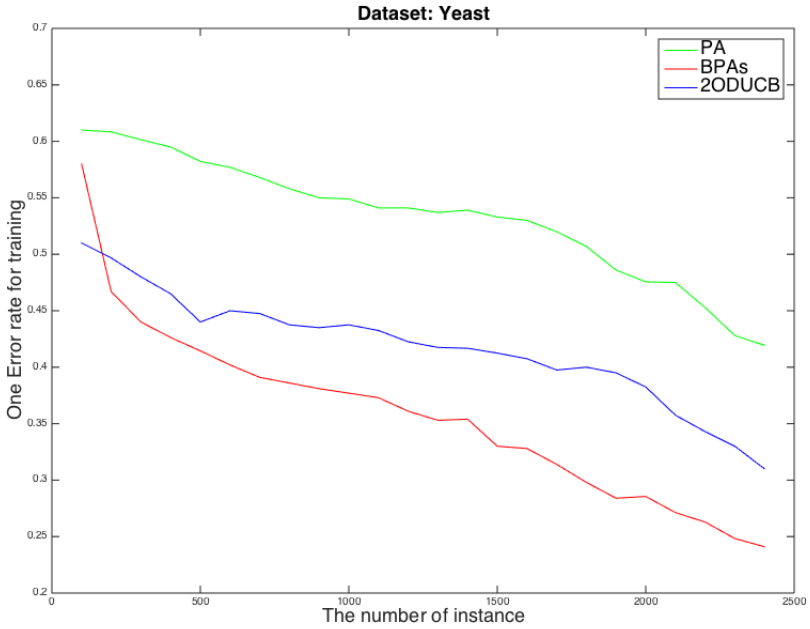

Figure 5.24: OneError of algorithms on Yeast

Chapter 6

Optimized Identification algorithm for ϵ-Pareto Front

In Chapter 4, we have introduced some definitions and methods about the Multi-Objective Multi-Armed Bandit problems. In this chapter, we continue to discuss this problem. We optimize the naive (ϵ,δ) algorithm of MAB to identify the ϵ-Pareto Front of MOMAB, and calculate two budget bounds in the pure exploration framework. These two bounds ensure the correctness and completeness of the ϵ-Pareto front.

6.1 Identification the ϵ-Pareto front

In this section, we propose a simple algorithm to find the Pareto front of Multi-Objective Multi-Armed Bandits. It extends the naive (ϵ,δ)-algorithm [48] to the Multi-Objective setting. The novel algorithm accurately identifies the optimal solutions with a tolerance δ. For the ϵ-dominance, we propose a novel fast algorithm to partially order the arm set, which is an improvement from [69].

There are several dominance relations that can partially order vectors in Multi-Objective Optimization. The Pareto dominance is the natural one, which has been introduced in Chapter 4. Here, we concentrates on MOO problems in the Bandit setting. Because of the stochastic characteristic of Bandit problems, the distribution probability vectors of arms are unknown to us. Therefore, we need to pull arms to approximate these vectors by empirical ones. Here, we use the method PAC [57] to calculate the number of

times that we have to pull each arm, so that we control the accuracy to be tolerable. The first related notion is ϵ-dominance (shown in many references [44, 46, 59, 66])

ϵ-Dominance. Let two vectors $\mathbf{a}$ and $\mathbf{b}$ be from the d-dimensional space $\mathscr{X} \subseteq \mathbb{R}^d$, if $\forall i \in [1,\ldots,d], a_i > b_i + \epsilon$ with $\epsilon \geqslant 0$, i.e. the vector $\mathbf{a}$ ϵ-dominates vector $\mathbf{b}$ denotes as $\mathbf{a} \succ_\epsilon \mathbf{b}$.

ϵ-Pareto Front. Given a set $\mathscr{A}^* \subseteq \mathscr{A}$. $\forall \mathbf{a} \in \mathscr{A}^*$, there is no vector $\mathbf{b} \in \mathscr{A}$, s.t. $\mathbf{b} \succ_\epsilon \mathbf{a}$. Therefore, this set $\mathscr{A}^*$ is called ϵ-Pareto front of set $\mathscr{A}$(denoted as $\mathscr{A}_\epsilon^*$). This set contains all arms which are not ϵ-dominated by any other arms of $\mathscr{A}$. The ϵ-Pareto front is formally given by

$$\mathscr{A}_\epsilon^* = \{\mathbf{a} \in \mathscr{A} | \nexists \mathbf{a}' \in \mathscr{A} \text{ s.t. } a_i' \succ_\epsilon a_i\}$$

Confidence Interval [11] means that the average reward within which the true expected reward falls with overwhelming probability.

The Hoeffding Inequality [58], is an useful tool to analyse the relationship between required number of samples and a confidence interval.

Theorem 6.1 (Hoeffding Inequlity)**.** *Let X be a set, and D be a fixed distribution on X. There is a mapping $f : x_i \to [a_i, b_i]$ for $i = [1,\ldots,n]$, where a_i and b_i are two real numbers and $a_i < b_i$. Let $x_1,\ldots,x_n \in X$ be drawn i.i.d. by distrbution D. We define the empirical mean of these variables by $\overline{X} = \frac{1}{n}(x_1 + \ldots, x_n)$.Then, the number of samples and tolerant error obeys the following inequality.*

$$\begin{aligned} \mathbb{P}[\overline{X} - \mathbb{E}[\overline{X}] \geqslant \epsilon] &\leqslant e^{-\frac{2\epsilon^2 n^2}{\sum_i (b_i - a_i)^2}} \\ \mathbb{P}[\overline{X} - \mathbb{E}[\overline{X}] \leqslant -\epsilon] &\leqslant e^{-\frac{2\epsilon^2 n^2}{\sum_i (b_i - a_i)^2}} \end{aligned} \tag{6.1}$$

Consider a set $\mathscr{A}$ with K arms, where $K \geqslant 2$. Each arm has D independent objectives. On round t, the player selects one arm a_k and receives a vector reward $\mathbf{r}_{k,t} = (r_{k,t}^1, r_{k,t}^2, \ldots, r_{k,t}^D) \in \{0,1\}^D$, which corresponds to a probability distribution vector $\mathbf{p}_k$, where $\mathbf{p}_k = (p_k^1, \ldots, p_k^D)$. By drawing arm a_k, the player estimates its empirical probability $\hat{\mathbf{p}}_k$. Let $T_k(N)$ be the number of arm a_k has been played during the first N pulls. The empirical probability $\hat{\mathbf{p}}_k$ is the current expected reward. It is estimated as $\hat{\mathbf{p}}_k = \sum_{t=1}^{T_k(N)} \mathbf{r}_{k,t} / T_k(N)$, where $\mathbf{r}_{k,t}$ is the reward vector of arm a_k on round t. Our target is to identify all optimal arms of set $\mathscr{A}$ with a minimal budget. In the algorithm, we propose pure exploration to obtain the information of all arms. It gives the ϵ-Pareto front with δ error probability. If all arms are pulled of the same times, it exists a lower bound and an upper bound for its complexity.

Lemma 6.1 (Lower bound)**.** *If we do not miss any optimal arm of the ϵ-Pareto front with δ-tolerant error probability, we need at least to pull n_{lower} times for each arm.*

$$n_{lower} = \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta}) \tag{6.2}$$

Proof. There are an optimal arm a and a non-optimal arm a' with distribution $\mathbf{p}_a$ and $\mathbf{p}_{a'}$, so $\mathbf{p}_a \succ \mathbf{p}_{a'}$. After n pulls, their empirical distribution are estimated as $\hat{\mathbf{p}}_a$ and $\hat{\mathbf{p}}_{a'}$. Let the arm a' ϵ-dominates the arm a ($\hat{\mathbf{p}}_{a'} \succ_\epsilon \hat{\mathbf{p}}_a$), iff $\forall i \in [1,\dots,D]$, $\hat{p}^i_{a'} > \hat{p}^i_a + \epsilon$. Here, we calculate the probability of $\hat{\mathbf{p}}_{a'} \succ_\epsilon \hat{\mathbf{p}}_a$.

$$\mathbb{P}(\hat{\mathbf{p}}_{a'} \succ_\epsilon \hat{\mathbf{p}}_a) = \mathbb{P}\left((\hat{p}^1_{a'} > \hat{p}^1_a + \epsilon) \cap \cdots \cap (\hat{p}^D_{a'} > \hat{p}^D_a + \epsilon)\right)$$

For all objectives of each vector are independent, there are the following deduction.

$$\begin{aligned}
\mathbb{P}(\hat{\mathbf{p}}_{a'} \succ_\epsilon \hat{\mathbf{p}}_a) &= \mathbb{P}\left(\hat{p}^1_{a'} > \hat{p}^1_a + \epsilon\right) * \cdots * \mathbb{P}\left(\hat{p}^D_{a'} > \hat{p}^D_a + \epsilon\right) \\
&= \prod_{i=1}^{D} \mathbb{P}(\hat{p}^i_{a'} > \hat{p}^i_a + \epsilon) \\
&\leqslant \prod_{i=1}^{D} \mathbb{P}(\hat{p}^i_{a'} > p^i_{a'} + \frac{\epsilon}{2} \text{ ou } \hat{p}^i_a < p^i_a - \frac{\epsilon}{2}) \\
&\leqslant \prod_{i=1}^{D} \left(\mathbb{P}(\hat{p}^i_{a'} > p^i_{a'} + \frac{\epsilon}{2}) + \mathbb{P}(\hat{p}^i_a < p^i_a - \frac{\epsilon}{2})\right) \\
&\leqslant \left(2\exp(-2(\frac{\epsilon}{2})^2 n)\right)^D
\end{aligned}$$

If we make an assumption that all arms may be this optimal arm a, the error probability for an arm a is bounded as $\mathbb{P}(\hat{\mathbf{p}}_{a'} \succ_\epsilon \hat{\mathbf{p}}_a) \leqslant \frac{\delta}{K}$. So,

$$\left(2\exp(-2(\frac{\epsilon}{2})^2 n)\right)^D = \frac{\delta}{K},$$

where n is the number that we pull arm a. For all arms, the number that they should be pulled

$$n_{lower} = \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta})$$

■

Lemma 6.2 (Upper bound)**.** *If we can eliminate all non-optimal arms from the ϵ-Pareto front with δ-tolerant error probability, we need pull n_{upper} times for each arm.*

$$n_{upper} = \frac{2K}{\epsilon^2}\ln\frac{2KD}{\delta} \tag{6.3}$$

Proof. There is a non-optimal arm a' and optimal arm a with distribution $\mathbf{p}_{a'}$ and $\mathbf{p}_a$ with $\mathbf{p}_a \succ \mathbf{p}_{a'}$. After n times pulling, the empirical distribution can be estimated as $\hat{\mathbf{p}}_{a'}$ and $\hat{\mathbf{p}}_a$. There is δ error probability to leave arm a' in ϵ-Pareto front. We here calculate the probability $\mathbb{P}(\hat{\mathbf{p}}_a \not\succ_\epsilon \hat{\mathbf{p}}_{a'})$.

$$\begin{aligned}
\mathbb{P}(\hat{\mathbf{p}}_a \not\succ_\epsilon \hat{\mathbf{p}}_{a'}) &\leqslant \mathbb{P}\left(\exists i \in [1,\dots,D], \hat{p}^i_{a'} + \epsilon > \hat{p}^i_a\right) \\
&\leqslant \mathbb{P}\left(\hat{p}^1_{a'} + \epsilon > \hat{p}^1_a \cup \cdots \cup \hat{p}^D_{a'} + \epsilon > \hat{p}^D_a\right) \\
&\leqslant \sum_{i=1}^{D} \mathbb{P}\left(\hat{p}^i_{a'} + \epsilon > \hat{p}^i_a\right) \\
&\leqslant \sum_{i=1}^{D} \left(\mathbb{P}(\hat{p}^i_{a'} > p^i_{a'} + \frac{\epsilon}{2} \text{ or } \hat{p}^i_a < p^i_a - \frac{\epsilon}{2})\right) \\
&\leqslant \sum_{i=1}^{D} 2 \cdot \exp\left(-2(\frac{\epsilon}{2})^2 n\right)
\end{aligned}$$

If each non-optimal arm a' has the probability $\frac{\delta}{K}$ that could appear in the ϵ-Pareto front. So that

$$2D \exp\left(-2(\frac{\epsilon}{2})^2 n\right) = \frac{\delta}{K}$$

For all arms, the number that they should be pulled

$$n_{upper} = \frac{2K}{\epsilon^2} \ln \frac{2KD}{\delta}.$$

■

ϵ-Pareto Identification algorithm

Firstly, we proposed the ϵ-Pareto Identification algorithm (see Algorithm 6.1). By improving the (ϵ, δ)-PAC algorithm of MAB, it can adapt Multi-Objective problems under Bandit environment and identify ϵ-Pareto front. By the lower bound, the algorithm can output an ϵ-Pareto which contains all true optimal arms with δ error probability. And we can eliminate all non-optimal arms from the ϵ-Pareto front after upper bound pulls. So, this algorithm starts with the lower bound. Every arm should be pulled n_{lower} times, then, we compare the empirical distribution of all arms with ϵ-dominance. After eliminating the ϵ-dominated arms, we pull the rest arms by $\left(n_{upper} - n_{lower}\right)$ times to identify the rest non-optimal arms in ϵ-Pareto front.

Algorithm 6.1 (ϵ-Pareto Identification algorithm)**.**

Initiate parameters ϵ and δ

for *Sample each arm $a \in \mathscr{A}$ $n_1 = \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D} \ln \frac{K}{\delta})$ times* ***do***

$\hat{\mathbf{p}}_a = \frac{1}{n_1}\sum_{t=1}^{n_1}\mathbf{r}_{a,t}$
end for
Identify the set $\mathscr{A}_\epsilon = \{a \in \mathscr{A} | \forall a' \in \mathscr{A}\, a' \not\succ_\epsilon a\}$.
for *Sample each arm* $a \in \mathscr{A}_\epsilon$ $n_2 = \frac{2K}{\epsilon^2}\ln\frac{2KD}{\delta} - \frac{2K}{\epsilon^2}(\ln 2 + \frac{1}{D}\ln\frac{K}{\delta})$ *times* ***do***
$\hat{\mathbf{p}}_a = \frac{1}{n_1+n_2}\sum_{t=1}^{n_1+n_2}\mathbf{r}_{a,t}$
end for
Output $\mathscr{A}_\epsilon^* = \{a \in \mathscr{A}_\epsilon | \nexists a' \in \mathscr{A}_\epsilon, a' \succ_\epsilon a\}$

In Algorithm 6.1, how to compare arms with ϵ-domiance is a crucial step that affects the final result. Under usual circumstances, it should compare all objectives of arms. When the number of arms or objectives increases, it will truly be a difficult task to complete. Therefore, we here propose a fast way to identify the ϵ-Pareto front, which is based on Kung's method [69] to find the maximal elements of a set of vectors. In [69], it proved that kung's method has lest complexity to find the maxima vector. For example, there are N D-dimensional vectors, if $D = 2,3$, its complexity is $O(N\log N)$, and $O\left(N\log N^{D-2}\right)$ for $D \geqslant 4$. Here, we provide an improvement of Kung's algorithm to find the maxima vectors for the ϵ-Pareto front. This method mostly reduces the complexity of searching and satisfies the requirement of comparing ϵ-dominance. It is classified by the number of objectives, e.g. $D = 2, D = 3$ or $D \geqslant 4$.

Look back to the model of MOMAB problem at the beginning of this chapter, this fast algorithm is to compare these empirical expected rewards vectors $\hat{\mathbf{p}}$. When $D = 2$, for K arms, their empirical vectors are $\hat{\mathbf{p}}_i \in \mathbb{R}^2$, with $i = [1,\ldots,K]$. We arrange these vectors by the first objective such that

$$\hat{p}^1_{\sigma(1)} \geqslant \hat{p}^1_{\sigma(2)} \geqslant \cdots \geqslant \hat{p}^1_{\sigma(K)}$$

where σ is a permutation function. It arranges these vectors by a descendant order.

Set parameters Indicator $i = 1$, $z^2 = \hat{p}^2_{\sigma(1)}$ and the ϵ-Pareto front $\mathscr{A}_{\epsilon 1} = \emptyset$. Make a sequence, compared $\hat{p}^2_{\sigma(i)}$ with z^2. If $\hat{p}^2_{\sigma(i)} + \epsilon > z^2$, this arm will be added to the ϵ-Pareto, and the reference z^2 will be replaced by the maximum of $\hat{p}^2_{\sigma(i)}$ and z^2. Move the indicator i to $i+1$. Until the indicator arrives to K, it output the set $\mathscr{A}_{\epsilon 1}$.

Then, this algorithm will start a new arrangement. It sorts descendant this vector set in second dimension

$$\hat{p}^2_{\sigma'(1)} \geqslant \hat{p}^2_{\sigma'(2)} \geqslant \cdots \geqslant \hat{p}^2_{\sigma'(K)},$$

and compared $\hat{\mathbf{p}}_{\sigma'(i)}$ with another reference z^1, where z^1 is initiated by $z^1 = \hat{p}^1_{\sigma'(1)}$, and it outputs a vector set $\mathscr{A}_{\epsilon 2}$. Finally, it merges these two sets $\mathscr{A}_{\epsilon 1}$and $\mathscr{A}_{\epsilon 2}$ to be ϵ-Pareto front (more details in Algorithm 6.2). The complexity of sorting algorithm is $O(K\log K)$ for one

dimension, and the comparison steps of another dimension requires $K-1$ times, so the complexity of this fast algorithm with $D=2$ is $O(K\log K)$.

Algorithm 6.2 (Fast ϵ-Pareto Finder ($D=2$))**.**

Require: *Set* $\mathscr{A}_{\epsilon 1}=\emptyset$, $\mathscr{A}_{\epsilon 2}=\emptyset$, $\mathbf{z}=[z^1,z^2]$;

Arrange all vectors $\hat{\mathbf{p}}\in\mathbb{R}^2$,
such that $\hat{p}^1_{\sigma(1)}>\hat{p}^1_{\sigma(2)}>\cdots>\hat{p}^1_{\sigma(N)}$;
for *each* $i=1,\ldots,N$ ***do***
  if $\hat{p}_{2,\sigma(i)}+\epsilon>z^2$ ***then***
    $\mathscr{A}_{\epsilon 1}=\mathscr{A}_{\epsilon 1}\cup\hat{\mathbf{p}}_{\sigma(i)}$
    $z^2=max(z^2,\hat{p}^2_{\sigma(i)})$
  end if
end for
Arrange all vectors $\hat{\mathbf{p}}\in\mathbb{R}^2$,
such that $\hat{p}^2_{\sigma'(1)}>\hat{p}^2_{\sigma'(2)}>\cdots>\hat{p}^2_{\sigma'(K)}$;
for *each* $i=1,\ldots,N$ ***do***
  if $\hat{p}^1_{\sigma'(i)}+\epsilon>z^1$ ***then***
    $\mathscr{A}_{\epsilon 2}=\mathscr{A}_{\epsilon 2}\cup\hat{\mathbf{p}}_{\sigma(i)}$
    $z^1=max(z^1,\hat{p}^1_{\sigma(i)})$
  end if
end for
Output $\mathscr{A}^*_\epsilon=\mathscr{A}_{\epsilon 1}\cup\mathscr{A}_{\epsilon 2}$

The fast algorithm for $D=3$ (see Algorithm 6.3) is an extension of the one with $D=2$. After arranging the first dimension, there are

$$\hat{p}^1_{\sigma(1)}\geqslant\hat{p}^1_{\sigma(2)}\geqslant\cdots\geqslant\hat{p}^1_{\sigma(K)}.$$

Set an ϵ-Pareto front $\mathscr{A}_{\epsilon 1}=\emptyset$. Firstly, let the indicator i equals to 1, $\mathscr{A}_{\epsilon 1}=\{\hat{\mathbf{p}}_{\sigma(1)}\}$. When the indicator goes to k, it arranges the Pareto set with the vector $\hat{\mathbf{p}}_{\sigma(k)}$ on the second dimension,

$$\hat{p}^2_{*,\sigma'(1)}\geqslant\hat{p}^2_{*,\sigma'(2)}\geqslant\cdots\geqslant\hat{p}^2_{\sigma'(k)}\geqslant\cdots\geqslant\hat{p}^2_{*,\sigma'(j)}\geqslant\cdots.$$

Here, the vectors $\hat{p}_{*,\sigma'(1)},\hat{p}_{*,\sigma'(2)},\ldots,\hat{p}_{*,\sigma'(j)},\ldots$ belong to the ϵ-Pareto front $\mathscr{A}_\epsilon$. By the new order σ', we find a vector $\hat{p}_{*,\sigma'(j)}$, if $\hat{p}^2_{\sigma'(k)}>\hat{p}^2_{*,\sigma'(j)}+\epsilon$, add $\hat{p}_{\sigma'(k)}$ to the ϵ-Pareto front. After that, it compares the third dimension of these vectors $\hat{p}^3_{*,\sigma'(j+1)},\hat{p}^3_{*,\sigma'(j+2)},\ldots$ with $\hat{p}^3_{\sigma'(k)}$, if $\hat{p}^3_{*,\sigma'(s)}+\epsilon<\hat{p}^3_{\sigma'(k)}$ with $s\geqslant j$. We will delete the vector $\hat{p}_{*,\sigma'(s)}$ from the ϵ-Pareto; otherwise,if in the ϵ-Pareto front, there is no such vector that $\hat{p}^2_{*,\sigma'(j)}+\epsilon<\hat{p}^2_{\sigma'(k)}$, we compared $\hat{p}^3_{\sigma'(k)}$ with $\left(max\{\hat{p}^3_{*,\sigma'(1)},\hat{p}^3_{*,\sigma'(2)},\ldots\}-\epsilon\right)$, if $\hat{p}^3_{\sigma'(k)}$ is much bigger, add $\hat{p}^3_{\sigma'(k)}$ to the ϵ-Pareto front.

Algorithm 6.3 (Fast ϵ-Pareto Finder ($D = 3$))**.**

Arrange all vectors $\hat{\mathbf{p}} \in \mathbb{R}^3$,
such that $\hat{p}^1_{\sigma(1)} > \hat{p}^1_{\sigma(2)} > \cdots > \hat{p}^1_{\sigma(K)}$
Set ϵ*-Pareto front* $\mathscr{A}_\epsilon = \hat{p}_{\sigma(1)}$, $max3 = \hat{p}^3_{\sigma(1)}$
for *each arm* $k = \sigma(1), \ldots, \sigma(K)$ ***do***
  Arrange all vectors of $\mathscr{A}_\epsilon$, $\hat{p}^2_{*,\sigma'(1)} \geqslant \hat{p}^2_{*,\sigma'(2)} \geqslant \cdots \geqslant \hat{p}^2_{*,\sigma'(j)} \geqslant \ldots$
  for *each arm* $\sigma'(j)$ *of* $\mathscr{A}_\epsilon$ ***do***
    if $\hat{p}^2_{\sigma'(k)} > \hat{p}^2_{*,\sigma'(j)} + \epsilon$ ***then***
      $\mathscr{A}_\epsilon = \mathscr{A}_\epsilon \cup \hat{p}_{\sigma'(k)}$
      if $\hat{p}^3_{\sigma'(k)} > \hat{p}^3_{*,\sigma'(j)} + \epsilon$ ***then***
        $\mathscr{A}_\epsilon = \mathscr{A}_\epsilon / \hat{p}_{*,\sigma'(j)}$
      end if
      $max3 = max(\hat{p}^3_{\sigma'(k)}, max3)$
    else
      if $\hat{p}^3_{\sigma'(k)} + \epsilon > max3$ ***then***
        $\mathscr{A}_\epsilon = \mathscr{A}_\epsilon \cup \hat{p}_{\sigma'(k)}$
        $max3 = max(\hat{p}^3_{\sigma'(k)} max3,)$
      end if
    end if
  end for
end for
Output $\mathscr{A}^*_\epsilon = \mathscr{A}_{\epsilon 1} \cup \mathscr{A}_{\epsilon 2}$

When $D \geqslant 4$, the problem could be decomposed into two subproblems and then combined the results together. The first step for this algorithm is also to arrange the vectors with first dimension descendent, like the following sequence

$$\hat{p}^1_{\sigma(1)} > \hat{p}^1_{\sigma(2)} > \ldots > \hat{p}^1_{\sigma(K)}$$

To segment the set into two sets by $A^* = (\hat{p}^1_{\sigma(1)}, \ldots, \hat{p}^1_{\sigma(s)})$ and $A_* = (\hat{p}^1_{\sigma(s+1)}, \ldots, \hat{p}^1_{\sigma(K)})$, with $s = \underset{i \in [1,\ldots,K]}{\arg\max} \lfloor (i \leqslant K/2) \rfloor$, s.t. $\hat{p}^1_{\sigma(s)} - \hat{p}^1_{\sigma(s+1)} > \epsilon$.

Then we find $\overline{A^*}$ and $\overline{A_*}$ the ϵ-Pareto front of $A^*$and A_*, respectively. Here, to identify the ϵ-Pareto front of subset A^* and A_* is with $(D-1)$-dimension. This recurrent process can be reused until that the subproblem can be resolved with the Fast ϵ-Pareto Finder $D = 2$ or $D = 3$. There is no doubt $\overline{A^*}$ is also the ϵ-Pareto for the set A_*. If we can find a set $T \subset \overline{A_*}$, who contains the arms are not ϵ-dominated by the elements of $\overline{A^*}$. So, the set T is also the ϵ-Pareto of set $\mathscr{A}$. Then, we combine the set $\overline{A^*}$ and T to output the final ϵ-Pareto.

Table 6.1: The parameters of algorithm "ϵ-Pareto Identification" and "Annealing Scalarized".

Algorithm	**Parameters**	
ϵ-Pareto Identification (10)	$\epsilon = 0.1$	$\delta = 0.05$
Annealing Scalarized (10)	$\epsilon_{decay} = 0.5$	$S = 10$
ϵ-Pareto Identification (30)	$\epsilon = 0.1$	$\delta = 0.1$
Annealing Scalarized (30)	$\epsilon_{decay} = 0.3$	$S = 30$

The principle of this algorithm is just to convert a problem from unknown to known, from complex to simple. Its process is shown in Algorithm 6.4. In this situation, the complexity of this algorithm is $O(N(\log N)^{D-2})$ (the demonstration refers to [69]).

Algorithm 6.4 (Fast ϵ-Pareto Finder ($D \geqslant 4$))**.**

Arrange all vectors $\hat{\mathbf{p}} \in \mathbb{R}^D$,
such that $\hat{p}^1_{\sigma(1)} > \hat{p}^1_{\sigma(2)} > \cdots > \hat{p}^1_{\sigma(K)}$;
Find $s = \underset{i \in [1,\ldots,K]}{arg\ max}\ \lfloor (i \leqslant K/2) \rfloor$, *s.t.* $\hat{p}^1_{\sigma(s)} - \hat{p}^1_{\sigma(s+1)} > \epsilon$
Set $A^* = (\hat{p}^1_{\sigma(1)}, \ldots, \hat{p}^1_{\sigma(s)})$ *and* $A_* = (\hat{p}^1_{\sigma(s+1)}, \ldots, \hat{p}^1_{\sigma(K)})$
Find respectively the ϵ*-Pareto front* $\overline{A^*}$ *for set* A^* *and* $\overline{A_*}$ *for set* A_*
Find a set $T = \{a \in \overline{A_*} | a' \not\succ_\epsilon a,\ where\ a' \in \overline{A^*}\}$
Output the ϵ*-Pareto front* $\mathscr{A}^*_\epsilon = \overline{A^*} \cup T$.

6.2 Experiments

Method. This experimentation is used to demonstrate the theoretical analysis. All points are generated at random in the range $[0,1]^2$. Datasets could be divided into three kinds: e.g. 'linear', 'convex' and 'concave'. For each data state, the points are bounded by a linear frontier, convex frontier and concave frontier, respectively. Let the coordinates of arm $\mathbf{x}$ be (x_1, x_2), which are considered as the expected reward of arm $\mathbf{x}$. During the experimental process, the reward for arm $\mathbf{x}$ is given randomly according its probability (x_1, x_2). Then we calculate the empirical expected probability by their rewards. Here, we compare our algorithm "ϵ-Pareto Identification" with the Annealing Scalarized approach [108], which is a UCB-scalarization method. **The goal of this experimentation is to prove that the Pareto front identified by our algorithm matches very well the real Pareto front, and we make less errors.** Some useful parameters of these algorithms are reported in Table 6.1

Graphic. From Fig. 6.1 to Fig. 6.3, each graph contains 10 points. And from Fig. 6.4 to Fig. 6.6, each graph contains 30 points. Each figure contains 4 graphs. According to the order, they are the original points; the points with the real Pareto front; ϵ-Pareto front of "ϵ-Pareto Identification"; the Pareto front of "Annealing Scalarized".

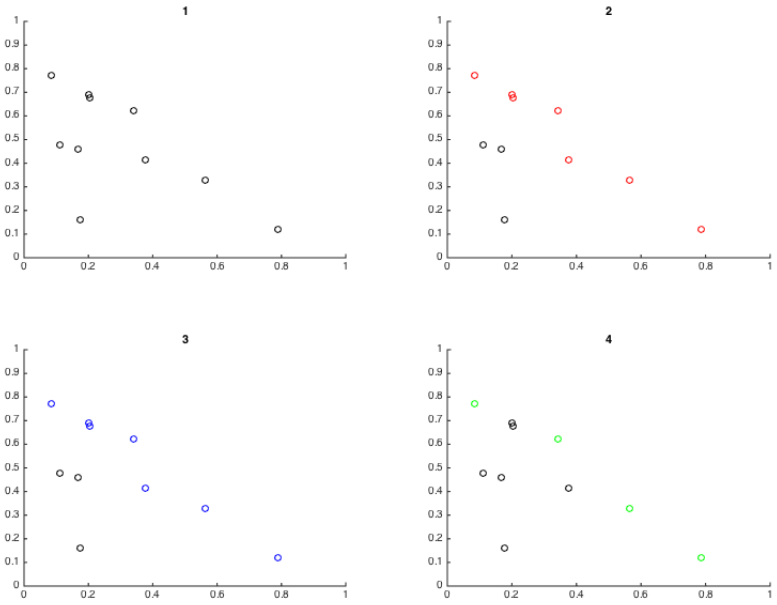

Figure 6.1: 10 points in linear state

Result. In Table 6.2, some place marked as "10/7″, it means that this algorithm identifies "10" arms as optimal arms in the Pareto front, "7" means there are only 7 arms in the real Pareto set. And from the results, we find both algorithms ϵ-Pareto Identification and Annealing Scalarized perform very well when there are only 10 points. However, when the number of points reaches 30, the scalarized algorithm shows its shortcomings that it is difficulty to find all optimal solutions with limited aggregation functions. Therefore, the ϵ-Pareto Identification algorithm does not have this kind problem. Under the theoretical analysis, if we can pull n_{lower} times for each arm, it can find all optimal arms. In fact, the empirical results can prove the theoretical analysis. Sometimes, the ϵ-Pareto Identification finds some wrong solutions, but this error is tolerant within the error probability δ.

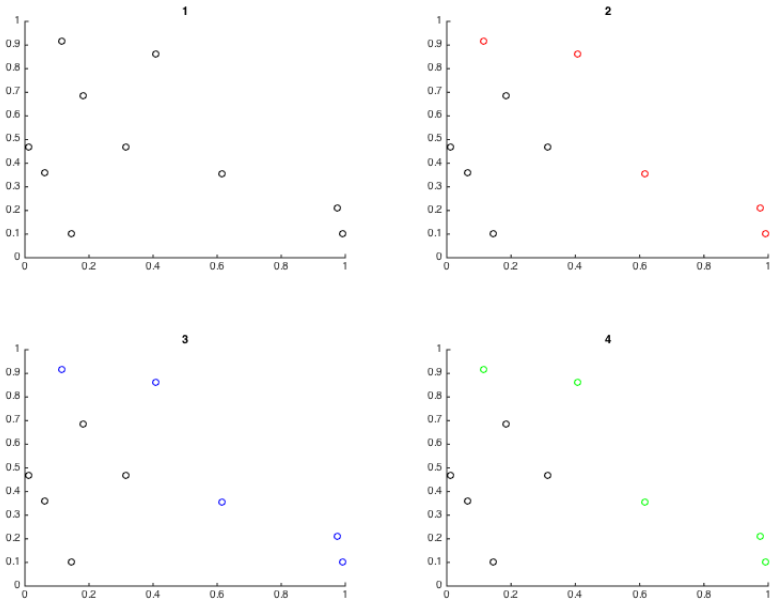

Figure 6.2: 10 points in convex state

Table 6.2: The results of algorithm "ϵ-Pareto Identification" and "Annealing Scalarized".

Algorithm	**Linear**	**Convex**	**Concave**
Real Pareto (10)	7	5	6
ϵ-Pareto Identification (10)	7/7	5/5	7/6
Annealing Scalarized (10)	4/4	5/5	3/3
Real Pareto (30)	7	7	12
ϵ-Pareto Identification (30)	8/7	7/7	12/12
Annealing Scalarized (30)	15/7	13/5	14/7

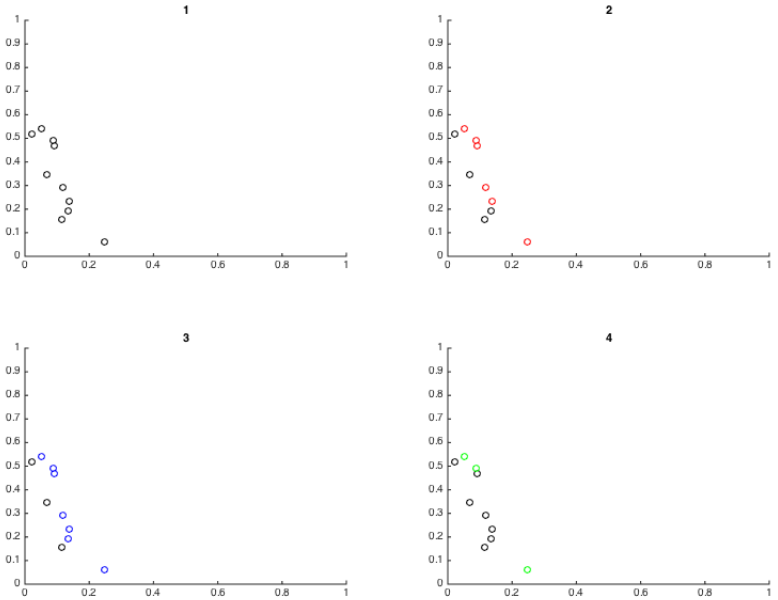

Figure 6.3: 10 points in concave state

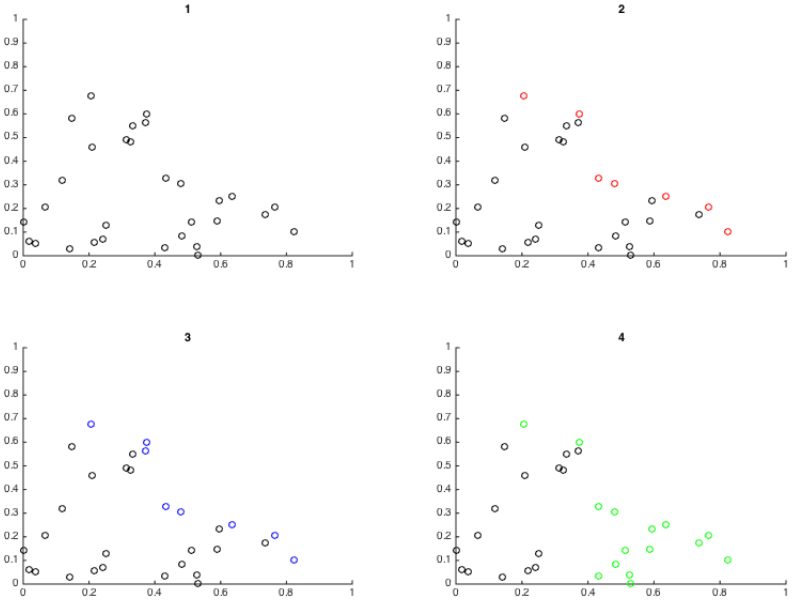

Figure 6.4: 30 points in linear state

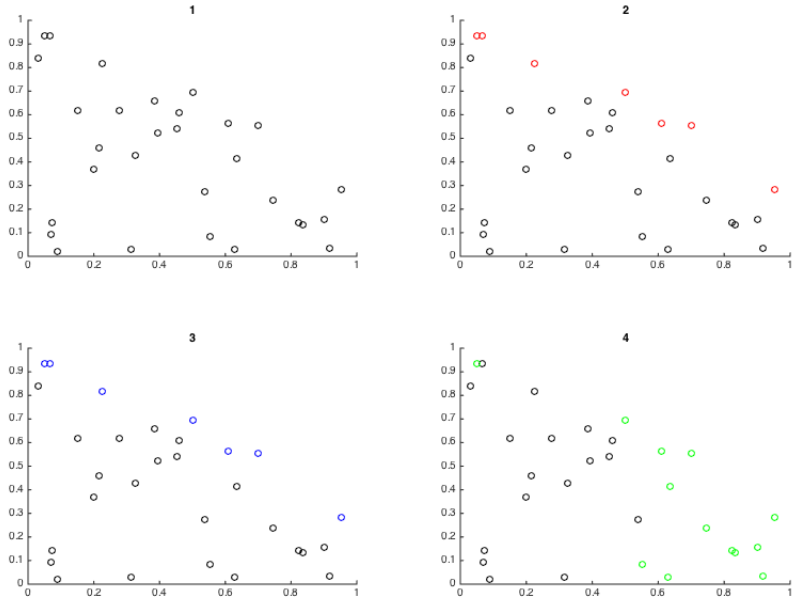

Figure 6.5: 30 points in convex state

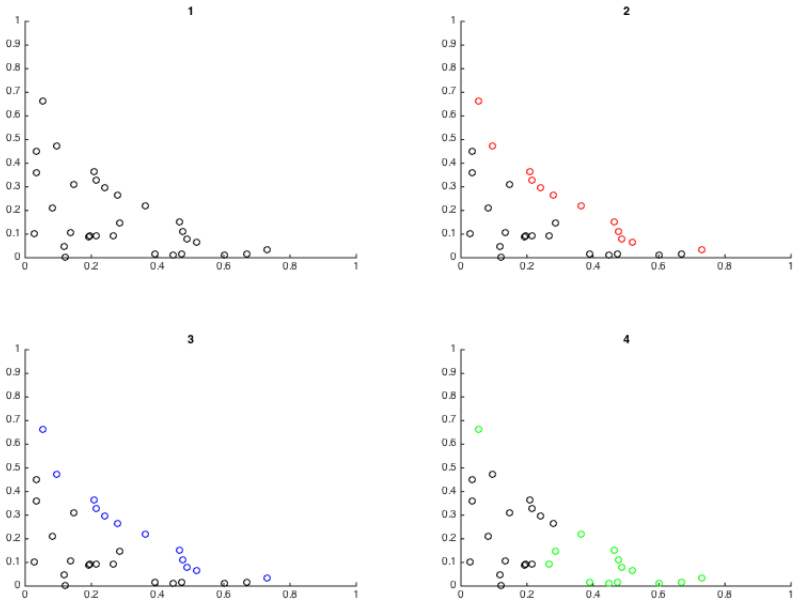

Figure 6.6: 30 points in concave state

Part III

Conclusion and Perspectives

Chapter 7

Conclusions

Bandit problems constitute a sequential dynamic allocation problem. The pulling agent has to explore its environment (i.e. the arms) to gather information on the one hand, and it has to exploit the collected clues to increase its rewards on the other hand. How to adequately balance the exploration phase and the exploitation phase is the crux of bandit problems and most of the efforts devoted by the research community from this field has focused on finding the right exploitation/exploration tradeoff. In this dissertation, we focus on investigating two specific bandit problems: the contextual bandit problems and the multi-objective bandit problems.

7.1 Summary of contributions

This dissertation provides two contributions. The first contribution is about the classification under partial supervision, which we encode as a contextual bandit problem with side information. This kind of problem is heavily studied by researchers working on social networks and recommendation systems. We provide a series of algorithms to solve the Bandit feedback problem that pertain to the Passive-Aggressive family of algorithms. We take advantage of its grounded foundations and we are able to show that our algorithms are much simpler to implement than state-of-the-are algorithms for bandit with partial feedback, and they yet achieve better performances of classification. The second contribution is about multi-objective multi-armed bandit problem (MOMAB). We propose an effective and theoretically motivated method to identify the Pareto front of arms. We in particular show that we can find all elements of the Pareto front with a minimal budget.

PBA provides an update item whose expectation equals to the PA one. It has two

variants, Simple PAB and Full PAB. The advantage of Simple PAB is its simplicity and perform very well on the linear separable dataset, but it is not fit to treat the non-separable datasets. Therefore, we add an anti-interference item, the expectation of this item being zero. It works well in stabilizing the classifiers, reducing the variance of the update and performing better than Simple PAB on non-separable dataset.

BPA, we design a hinge-loss function to adapt the problem of classification with Bandit feedback. The BPA algorithm is grounded on this loss function. By a theoretical analysis, we find the squared loss of BPA to be bounded by the same bound than the Passive-Aggressive algorithm, which works in supervised setting. From an empirical perspective, we performed numerical experiments using this algorithm and some others. While displaying a lesser complexity, the performance of our algorithm is close to the performance of the second order perceptron Bandit algorithm for all datasets. This performance is moreover often better than the performance of supervised classifiers on some non-separable datasets.

Kernel BPA and **Kernel SGD with Bandit loss** are two algorithms adapted to the RKHS framework. Those two algorithms are devoted to learning with Bandit feedback on non-linear datasets. In section 5.3, we provide details on those approaches. The first one is a direct derivation of the BPA algorithm, i.e. it directly maps the product of two vectors into RKHS. This method has good accuracy, but the complexity of classifiers increases linearly along with the proportion of non-separable instances. This brings some trouble to the computational efficiency. Referring to Stochastic Gradient Descent [21] and Online Kernel method [64], we optimized the kernel classifiers by Bandit loss function. Though its accuracy do not reach the level of **Kernel BPA**, its computational complexity is stable. It can easily face to any kind of datasets.

Multi-label BPA, for Multi-label Classification, we also propose a novel algorithm. We transfer the Multi-label classification into multiple Multi-class classifications, at the same time, it trains the multiple classifiers by the BPA algorithms. This algorithm, both from the theoretical analysis and empirical result, is found to perform very well.

ϵ-Pareto Identification. In Chapter 6, we propose a method to treat the optimal solutions of MOMAB problem. Different to Multi-Objective Optimization, MOMAB approximates the expected value of each options by calculating the rewards. So the values of each option has some stochastic difference to the true one. Therefore, we replace the Pareto front by the ϵ-Pareto front. Through a theoretical analysis, we get a bound, assuming that all optimal arms can be posed in the ϵ-Pareto front if the number of each arm pulling reaches this bound. Otherwise, there is another bound to delete all non-optimal arms from the ϵ-Pareto front. By the rule of these two bounds, we propose an algorithm to identify the ϵ-Pareto front. At the same time, according to the Algorithm of Kung [69], we design a fast

algorithm to compare the ϵ-dominance across all arms. When there is a huge set of arms, this method can locate the ϵ-Pareto front quickly.

7.2 Research in the future

Several limitations of our results need to be adressed in the future. Regarding the contextual with side information, we should put a stronger focus on the tradeoff between exploration and exploitation. Our algorithms is based on the ϵ-greedy strategy (see in section 2.3.4) to keep the balance between exploration and exploitation. At each round, it samples the labels with a fixed probability $\frac{\epsilon}{K}$ to explore, and $1-\epsilon$ to exploit. If we always have some probability to explore, this results in wasting the budgets, after the classifier has performed very well. In Figure 5.9, 5.10 and 5.11, we observe that the result of algorithm BPA has less difference between the different of value ϵ. This result naturally suggests us to use variable ε across the learning session, allowing more exploration in the first trials and more exploitation in the last ones. Next step, we can try the dynamic ϵ-greedy [11], that parameter ϵ attenuate according to $1/t$. Otherwise, there is another limitation for our algorithms, it is how to choose the parameters, e.g. in algorithm PAB, there are C for the soft-margin, ϵ for ϵ-greedy; in KBPA, the parameters of kernel function. Generally, we need to take a cross-validation to determine these parameters. However, when an algorithm has two or more parameters to choose, the comparing work becomes very difficult.

The second aspect is about the Pareto front of MOMAB problem. At present, we use pure exploration to identify the optimal arms for getting the Pareto front. This way can ensure the integrity of the Pareto set, however it wastes some budgets compared with other methods. So next step, we need to continue to optimize this method.

The work of this dissertation opens several theoretical and applicative perspectives for the future. From the applicative perspective, this work has a very broad prospect. We already mentioned the applications in social networks and recommendation systems. Another example is decision making in labor markets [109]. Problems like Internet packet routing or smart grid electricity distribution can also be considered, where the Multi-objective bandit framework may provide an effective framework for routing optimization. Bandit framework can also combine with other machine learning frameworks, e.g. deep learning, active learning, semi-supervised learning, for they share a common principles.

Finally, those three years of PhD preparation, not only influenced my view on research and work, but also changed my attitude toward life and philosophy.

Appendix A

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* $k \in \{1,\ldots,K\}$

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

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

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

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* $k = 1,\ldots,K$.

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* $k = 1,\ldots,K$.

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

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,. . . ,* 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,. . . , 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:

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

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

$$\hat{y}_t = arg\ max_i(\hat{\Delta}'_{i,t} + \epsilon_{i,t}), where \epsilon^2_{i,t} = \left(2\mathbf{x}_t^T A^{-1}_{i,t} \mathbf{x}_t\right) \times \eta_t$$
$$and \eta_t = \frac{1}{2}(1+\alpha)^2 \| U \|_2^2 + \frac{(1+\alpha)^2}{2} \sum_{s=1}^{t-1} \mathbf{x}_s^T A^{-1}_{\hat{y}_s, s} \mathbf{x}_s + 9(1+\alpha)^2 \log \frac{t+4}{\delta}$$

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 $P := (P = \{z \in P | y \not\succeq z\}) \cup \{y\}$

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 $P := (P = \{z \in P | b(y) \not\succeq b(z)\}) \cup \{y\}$

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

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] J. ALCALA-FDEZ, L. SANCHEZ, S. GARCIA, M. DEL JESUS, S. VENTURA, J. GARRELL, J. OTERO, C. ROMERO, J. BACARDIT, V. RIVAS, J. FERNANDEZ, AND F.HERRERA, *Keel: A software tool to assess evolutionary algorithms to data mining problems*, vol. 13 of Soft Computing, 2009, pp. 307–318.

[6] F. ALIMOGLU AND E. ALPAYDIN, *Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition*, in Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96, 1996.

[7] F. ALIMOGLU, D. DOC, E. ALPAYDIN, AND Y. DENIZHAN, *Combining multiple classifiers for pen-based handwritten digit recognition*, (1996).

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

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

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

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

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

[13] 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.

[14] 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.

[15] D. BAATAR AND M. M. WIECEK, *Advancing equitability in multiobjective programming*, Computers & Mathematics with Applications, 52 (2006), pp. 225–234.

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

[17] P. L. BARTLETT AND M. H. WEGKAMP, *Classification with a reject option using a hinge loss*, The Journal of Machine Learning Research, 9 (2008), pp. 1823–1840.

[18] R. BEKKERMAN AND M. SCHOLZ, *Data weaving: Scaling up the state-of-the-art in data clustering*, no. 1083–1092, In Proceedings of CIKM, 2008.

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

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

[21] L. BOTTOU, *Large-scale machine learning with stochastic gradient descent*, in Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[22] S. BOYD AND L. VANDENBERGHE, *Convex optimization*, (2004).

[23] 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.

[24] S. BUBECK, R. MUNOS, AND G. STOLTZ, *Pure exploration in multi-armed bandits problems*, in Algorithmic Learning Theory, Springer, 2009, pp. 23–37.

[25] ——, *Pure exploration in finitely-armed and continuous-armed bandits*, Theoretical Computer Science, 412 (2011), pp. 1832–1852.

[26] C. CARLSSON AND R. FULLÉR, *Owa operators for decision support*, Proceedings of EUFIT, 97 (1997), pp. 1539–1544.

[27] A. CARPENTIER AND R. MUNOS, *Bandit theory meets compressed sensing for high dimensional stochastic linear bandit*, arXiv preprint arXiv:1205.4094, (2012).

[28] N. CESA-BIANCHI, A. CONCONI, AND C. GENTILE, *A second-order perceptron algorithm*, SIAM Journal on Computing, 34 (2005), pp. 640–668.

[29] N. CESA-BIANCHI AND P. FISCHER, *Finite-time regret bounds for the multiarmed bandit problem.*, in ICML, Citeseer, 1998, pp. 100–108.

[30] O. CHAPELLE AND L. LI, *An empirical evaluation of thompson sampling*, in Advances in neural information processing systems, 2011, pp. 2249–2257.

[31] 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.

[32] F. O. CLAUDIO GENTILE, *On multilabel classification and ranking with bandit feedback*, Journal of Machine Learning Research, (2014).

[33] 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.

[34] K. CRAMMER AND C. GENTILE, *Multiclass classification with bandit feedback using adaptive regularization*, Mach Learn, 90 (2013), pp. 347–383.

[35] K. CRAMMER AND Y. SINGER, *Ultraconservative online algorithms for multiclass problems*, The Journal of Machine Learning Research, 3 (2003), pp. 951–991.

[36] E. DAUCÉ, T. PROIX, AND L. RALAIVOLA, *Fast online adaptivity with policy gradient: example of the BCI "p300"-speller*, in 21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013, 2013.

[37] 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.

[38] K. DEB, *Multi-objective optimization using evolutionary algorithms*, vol. 16, John Wiley & Sons, 2001.

[39] K. DEB, S. AGRAWAL, A. PRATAP, AND T. MEYARIVAN, *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii*, Lecture notes in computer science, 1917 (2000), pp. 849–858.

[40] K. DEMBCZYŃSKI, W. WAEGEMAN, W. CHENG, AND E. HÜLLERMEIER, *On label dependence and loss minimization in multi-label classification*, Machine Learning, 88 (2012), pp. 5–45.

[41] M. DREDZE, K. CRAMMER, AND F. PEREIRA, *Confidence-weighted linear classification*, in Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 264–271.

[42] 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.

[43] M. M. DRUGAN, *Linear scalarization for pareto front identification in stochastic environments*, in Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 156–171.

[44] M. M. DRUGAN, A. BE, AND A. NOWÉ, *ϵ-approximate pareto optimal set of arms identification in multi-objective multi-armed bandits*, in BENELEARN 2014-23th annual Belgian-Dutch conference on Artificial Intelligence, 2014.

[45] 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.

[46] G. EICHFELDER, *An adaptive scalarization method in multiobjective optimization*, SIAM Journal on Optimization, 19 (2009), pp. 1694–1718.

[47] A. ELISSEEFF AND J. WESTON, *A kernel method for multi-labelled classification*, in Advances in neural information processing systems, 2001, pp. 681–687.

[48] 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.

[49] W. GAO AND Z.-H. ZHOU, *On the consistency of multi-label learning*, Artificial Intelligence, 199 (2013), pp. 22–44.

[50] A. GARIVIER AND E. MOULINES, *On upper-confidence bound policies for non-stationary bandit problems*, Arxiv preprint, (2008).

[51] C. GENTILE AND M. K. WARMUTH, *Linear hinge loss and average margin*, in NIPS, vol. 11, 1998, pp. 225–231.

[52] O. GIEL AND P. K. LEHRE, *On the effect of populations in evolutionary multi-objective optimisation**, Evolutionary Computation, 18 (2010), pp. 335–356.

[53] J. GITTINS AND D. JONES, *A Dynamic Allocation Index for the Sequential Allocation of Experiments*, Progress in Statistics, 1974.

[54] J. C. GITTINS, *Bandit processes and dynamic allocation indices*, Journal of the Royal Statistical Society. Series B (Methodological), (1979), pp. 148–177.

[55] J. C. GITTINS AND D. M. JONES, *A dynamic allocation index for the discounted multiarmed bandit problem*, Biometrika, 66 (1979), pp. 561–565.

[56] GOOGLE, *Google analytic service*, https://www.google.com/analytics/, (2015).

[57] D. HAUSSLER, *Probably approximately correct learning*, University of California, Santa Cruz, Computer Research Laboratory, 1990.

[58] W. HOEFFDING, *Probability inequalities for sums of bounded random variables*, Journal of the American statistical association, 58 (1963), pp. 13–30.

[59] 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.

[60] A. F. J. ALCALA-FDEZ, J. LUENGO, J. DERRAC, S. GARCIA, L. SANCHEZ, AND F. HERRERA, *Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework*, vol. 17 of Journal of Multiple-Valued Logic and Soft Computing, 2011, pp. 255–287.

[61] L. P. KAELBLING, M. L. LITTMAN, AND A. W. MOORE, *Reinforcement learning: A survey*, Journal of artificial intelligence research, (1996), pp. 237–285.

[62] 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.

[63] E. KAUFMANN, N. KORDA, AND R. MUNOS, *Thompson sampling: An asymptotically optimal finite-time analysis*, in Algorithmic Learning Theory, Springer, 2012, pp. 199–213.

[64] J. KIVINEN, A. J. SMOLA, AND R. C. WILLIAMSON, *Online learning with kernels*, Signal Processing, IEEE Transactions on, 52 (2004), pp. 2165–2176.

[65] 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.

[66] J. B. KOLLAT, P. REED, AND J. KASPRZYK, *A new epsilon-dominance hierarchical bayesian optimization algorithm for large multiobjective monitoring network design problems*, Advances in Water Resources, 31 (2008), pp. 828–845.

[67] 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.

[68] M. KOSTREVA, W. OGRYCZAK, AND A. WIERZBICKI, *Equitable aggregations and multiple criteria analysis*, European Journal of Operational Research, 158 (2004), pp. 362–377.

[69] H.-T. KUNG, F. LUCCIO, AND F. P. PREPARATA, *On finding the maxima of a set of vectors*, Journal of the ACM (JACM), 22 (1975), pp. 469–476.

[70] T. L. LAI AND H. ROBBINS, *Asymptotically efficient adaptive allocation rules*, Advances in applied mathematics, 6 (1985), pp. 4–22.

[71] 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.

[72] J. LEGRIEL, C. LE GUERNIC, S. COTTON, AND O. MALER, *Approximating the pareto front of multi-criteria optimization problems.*, in TACAS, Springer, 2010, pp. 69–83.

[73] D. LEWIS, Y. YANG, T. ROSE, AND F. LI, *Rcv1: A new benchmark collection for text categorization research*, vol. 5 of JMLR, 2004, pp. 361–397.

[74] M. LICHMAN, *UCI machine learning repository*, 2013.

[75] N. LITTLESTONE, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*, Machine learning, 2 (1988), pp. 285–318.

[76] X. LIU, *Some properties of the weighted owa operator*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 36 (2006), pp. 118–127.

[77] 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.

[78] R. D. LUCE, *Individual choice behavior: A theoretical analysis*, Courier Corporation, 1959.

[79] 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.

[80] 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.

[81] C. MCFARLAND, *Experiment!: Website conversion rate optimization with A/B and multivariate testing*, New Riders, 2012.

[82] K. MIETTINEN, *Nonlinear multiobjective optimization*, vol. 12, Springer Science & Business Media, 2012.

[83] M. L. MINSKY AND S. A. PAPERT, *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*, MIT press Boston, MA:, 1987.

[84] V. PARETO, *Cours d'economies politique, volume ? and ?*, F Rouge, Lausanne, (1896).

[85] J. READ, *A pruned problem transformation method for multi-label classification*, in Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008), vol. 143150, 2008.

[86] H. ROBBINS, *Some aspects of the sequential design of experiments*, Bulleting of the American Mathematical Society, 58 (1952), pp. 527–535.

[87] F. ROSENBLATT, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), p. 386.

[88] L. C. S. SCHUURMANS AND S. CAELLI, *Implicit online learning with kernels*, Advances in neural information processing systems, 19 (2007), p. 249.

[89] S. L. SCOTT, *A modern bayesian look at the multi-armed bandit*, Applied Stochastic Models in Business and Industry, 26 (2010), pp. 639–658.

[90] O. SHAMIR, *Fundamental limits of online and distributed algorithms for statistical learning and estimation*, in Advances in Neural Information Processing Systems, 2014, pp. 163–171.

[91] S. SHAN AND G. G. WANG, *An efficient pareto set identification approach for multi-objective optimization on black-box functions*, Journal of Mechanical Design, 127 (2005), pp. 866–874.

[92] K. SLAVAKIS, S. THEODORIDIS, AND I. YAMADA, *Online kernel-based classification using adaptive projection algorithms*, Signal Processing, IEEE Transactions on, 56 (2008), pp. 2781–2796.

[93] M. S. SOROWER, *A literature survey on algorithms for multi-label learning*, Oregon State University, Corvallis, (2010).

[94] 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.

[95] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*, vol. 1, MIT press Cambridge, 1998.

[96] 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.

[97] 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.

[98] J. N. TSITSIKLIS, *A short proof of the gittins index theorem*, The Annals of Applied Probability, (1994), pp. 194–199.

[99] V. N. VAPNIK AND V. VAPNIK, *Statistical learning theory*, vol. 1, Wiley New York, 1998.

[100] 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.

[101] 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.

[102] R. WEBER ET AL., *On the gittins index for multiarmed bandits*, The Annals of Applied Probability, 2 (1992), pp. 1024–1033.

[103] P. WHITTLE, *Multi-armed bandits and the gittins index*, Journal of the Royal Statistical Society. Series B (Methodological), (1980), pp. 143–149.

[104] W.OGRYCZAK AND T. TRZASKALIK, *Equity, fairness and multicriteria optimization*, Multiple Criteria Decision Making'05, (2005), pp. 185–199.

[105] R. R. YAGER, *On ordered weighted averaging aggregation operators in multicriteria decisionmaking*, Systems, Man and Cybernetics, IEEE Transactions on, 18 (1988), pp. 183–190.

[106] ——, *Families of owa operators*, Fuzzy sets and systems, 59 (1993), pp. 125–148.

[107] 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.

[108] S. Q. YAHYAA, M. M. DRUGAN, AND B. MANDERICK, *Annealing linear scalarized based multi-objective multi-armed bandit algorithm*.

[109] E. YASHIV, *Labor search and matching in macroeconomics*, European Economic Review, 51 (2007), pp. 1859–1895.

[110] H. ZHONG AND E. DAUCÉ, *Passive-aggressive bounds in bandit feedback classification*, in Proceedings of the ECMLPKDD 2015 Doctoral Consortium, J. Hollmén and P. Papapetrou, eds., Aalto University publication series SCIENCE + TECHNOLOGY, 12/2015, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2015, 2015, pp. 255–264.

[111] H. ZHONG, E. DAUCÉ, AND L. RALAIVOLA, *Online multiclass learning with "bandit" feedback under a passive-aggressive approach*, in proceedings of European Symposium on Artificial Neural Networks Computational Intelligence and Machine Learning (ESANN), 2015, pp. 403–408.

[112] 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.