Batched Multi-armed Bandits Problem

Analisi critica

Docente:	Francesco Trovò
Studente:	**Andi Ferhati**
Matricola:	1012310

Materia:	Corso di Intelligenza Artificiale
Appello:	Sessione estiva, 19 Giugno 2020
Anno:	Anno accad. 2019-2020 - I anno
Corso:	CdL Ing. Informatica, Magistrale
Università:	Università degli Studi di Bergamo

Indice

Elenco delle figure

Elenco dei frammenti di codice

Elenco dei teoremi

Acronimi

ABSE Adaptively Binned Successive Elimination 23

B2aB Batched 2-armed Bandit 12

BaSE Batched Successive Elimination v, vii, ix, 5, 25–28, 35, 42, 43, 53–55, 74, 75, 77, 78

BMaB Batched Multi-armed Bandit 5, 6, 9, 19, 23, 26, 30, 55

ETC Explore Than Commit v, 35, 42, 43, 53, 54, 75

i.i.d. indipendenti e identicamente distribuiti 6, 16

INF Implicitly Normalized Forecaster 17

MaB Multi-armed Bandit 5, 7, 11, 14–19, 22, 23, 25, 77, 78

MbKaB M-batched K-armed Bandit ix, 8, 9, 28

NIPS Neural Information Processing Systems 3

NPV Net Present Value 15

SE Successive Elimination 23, 26

sse se e solo se 25

TaSB Two-armed Stochastic Bandit 5

UCB Upper Confidence Bound xiii, 18, 26

UCB1 Upper Confidence Bound 1 16, 17, 35, 53

Nomenclatura

Lettere greche

Δ_i il *gap* tra l'*arm* i e l'*arm* migliore

γ un parametro di *tuning* associato al *Upper Confidence Bound (UCB)*

μ la media della distribuzione gaussiana dei *rewards*

$\mu^\star$ il valore atteso del *reward* per l'*arm* migliore

ν una distribuzione gaussiana dei *rewards*

π una *policy* di campionamento

$\Pi_{M,T}$ l'insieme delle *policy* con M *batches* e l'orizzonte temporale T

Lettere latine capitali

$\mathcal{A}$ l'insieme di *arms* attivi

$\mathcal{I}$ l'insieme di *arms*

K il numero di *arms*

M il numero di *batches*

P_i^t la distribuzione delle osservazioni disponibili al momento t sotto P_i

$P_{j,k}(A)$ la probabilità dell'evento A data la distribuzione vera del *reward* $P_{j,k}$ e la *policy* π

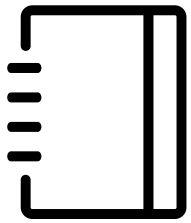
$R^t(\pi)$ il *regret* istantaneo incorso dalla *policy* π_t al momento t

$R^\star_{\text{min}-\text{max}}$ *minimax regret* sotto l'impostazione *batched*

$R^\star_{\text{pro-dep}}$ il *regret dipendente dal problema* sotto l'impostazione *batched*

R_T il *regret* cumulativo (pseudo-*regret* o semplicemente *regret*)

T l'orizzonte temporale



Capitolo 1

Introduzione

Logo in Copertina[1]
Icona del Capitolo 1[2]

1.1 init

In questo elaborato si analizza il *paper* «Batched Multi-armed Bandits Problem»[3] degli autori Gao, Han, Ren e Zhou. Tale *articolo* fa parte dei *proceedings* della conferenza *Neural Information Processing Systems (NIPS)* del 2019.

1.2 Struttura e sezioni dell'elaborato

Nel Capitolo 2 - Contesto, viene descritto il contesto in cui si pone l'oggetto di analisi, quali sono gli scenari applicativi a cui può essere applicato e quale è il problema analizzato dal *paper*.

Nel Capitolo 3 - Studi correlati, si elencano, descrivono e commentano i lavori precedenti in merito allo stesso problema o studi di problemi simili. I lavori citati nel o correlati al *paper* oggetto di studio vengono elencati in sezione 3.2 - Elenco degli studi citati nel *paper* originale;

Nel Capitolo 4 - Descrizione, viene spiegato il funzionamento dell'algoritmo, presentato nel *paper* sotto forma di pseudocodice, e una descrizione scritta dei vari aspetti analizzati nel *paper*. Inoltre, vengono descritte e commentare le proprietà teoriche presentate.

[1] Università degli Studi di Bergamo. *Logo UniBG*. Online. Mag. 2020. URL: https://unibg.it/.
[2] UXWing. *Icons*. Online. Mag. 2020. URL: https://uxwing.com/.
[3] Zijun Gao, Yanjun Han, Zhimei Ren e Zhengqing Zhou. «Batched Multi-armed Bandits Problem». Inglese. In: *Advances in Neural Information Processing Systems 32*. A cura di H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox e R. Garnett. Documento locale in pdf. Curran Associates, Inc., 2019, pagine 503–513. URL: http://papers.nips.cc/paper/8341-batched-multi-armed-bandits-problem.pdf.

Nel Capitolo 5 - Esperimenti, vengono descritti gli esperimenti fatti su dei dati reali utilizzando il metodo analizzato. In particolare:

- si illustrano gli esperimenti presentati nel *paper*;
- si ripetono gli stessi esperimenti presentati nel *paper* in codice in Matlab e Python, si osservano e commentano i risultati;
- si è scelto un *dataset* da fonte aperta *online* e si è applicato a esso il metodo descritto. Si cerca di simulare gli esperimenti effettuati su dati generati a *random* questa volta eseguendoli su dati reali. Si osservano e commentano i risultati.

Nel Capitolo 6 - Conclusioni, sono elencate le conclusioni sulle proprietà teoriche e sperimentali del metodo analizzato nel *paper*. Inoltre, in questo capitolo sono elencati alcuni possibili usi nella vita reale e in ambito *machine learning* delle *policies* proposte nello studio.

Nel Capitolo 7 - Considerazioni personali, si esprime brevemente un pensiero personalo sul lavoro di preparazione di questo elaborato per poi concludere con i *tools* e librerie usati.

Seguono alla fine il Glossario dei termini e i Riferimenti bibliografici dove sono elencati tutti i lavori citati nell'elaborato.

Capitolo 2

Contesto

Icona del Capitolo 2[2]

2.1 Oggetto di studio

L'oggetto di studio di questo elaborato è il *paper* di Gao, Han, Ren e Zhou sul problema del *Multi-armed Bandit (MaB)* a impostazione *batched* detto *Batched Multi-armed Bandit (BMaB)*, dove la *policy* impiegata è quella di suddividere i dati in un piccolo numero di *batches*[3]. La motivazione alla base di tale studio è che mentre il *minimax regret* per il problema del *Two-armed Stochastic Bandit (TaSB)* è stato caratterizzato a pieno in «Batched bandit problems»[4] da Perchet, Rigollet, Chassang e Snowberg, l'effetto del numero degli *arms* nel *regret* del caso *multi-armed* è ancora un argomento aperto. Inoltre, rimane ancora inesplorata la domanda se dimensioni di *batch* scelti in modo adattivo aiutano a ridurre il *regret*. Nel *paper* oggetto di analisi si propone la *policy Batched Successive Elimination (BaSE)* per ottenere *rate-optimal regrets* (all'interno di fattori logaritmici) per il *Batched Multi-armed Bandit (BMaB)*, con *matching lower bounds* anche se le dimensioni dei *batches* sono determinate in modo adattivo.

2.2 Panoramica generale

Il *batch learning* e l'*online learning* sono due aspetti importanti di *machine learning*. Nel *batch learning* il *learner* è un osservatore passivo di una data collezione di dati da cui imparare. Nell'*online learning* invece può attivamente determinare il processo di acquisizione dei dati.

[4] Vianney Perchet, Philippe Rigollet, Sylvain Chassang e Erik Snowberg. «Batched bandit problems». Inglese. In: *Annals of Statistics* 44.2 (apr. 2016). Documento locale in pdf, pagine 660–681. DOI: 10.1214/15-AOS1381. URL: https://projecteuclid.org/euclid.aos/1458245731.

Recentemente la combinazione di queste procedure di *learning* ha visto un incremento in molte applicazioni, dove il *querying* attivo dei dati è possibile però limitato a un numero fisso di *round* di interazioni. Per esempio, nei test clinici[5][6], i dati arrivano in *batches* dove i gruppi di pazienti sono trattati simultaneamente per delineare il prossimo test. Nell'ambito del *crowdsourcing*[7], alla *crowd* è necessario un po' di tempo per rispondere alle domande poste, quindi i vincoli sul tempo totale impongono restrizioni sul numero di *round* di interazioni. Problemi simili si vedono anche nel *marketing*[8] e nel *simulation selection problem*[9].

Nel *paper* analizzato si studia l'influenza dei vincoli sui *rounds* nella *learning performance* mediante il *Batched Multi-armed Bandit (BMaB)*. Sia $\mathcal{I} = \{1, 2, \cdots, K\}$ un *set* dato di $K \geq 2$ *arms* di un *stochastic bandit*, dove tirate successive di un *arm* $i \in \mathcal{I}$ produce *rewards* che sono campioni indipendenti e identicamente distribuiti (i.i.d.) della distribuzione $\nu^{(i)}$ con media $\mu^{(i)}$. Nel *paper* si assume che il *reward* segua una distribuzione gaussiana, i.e., $\nu^{(i)} = \mathcal{N}\left(\mu^{(i)}, 1\right)$, dove generalizzazioni a generici *reward* sub-gaussiani e varianze sono immediate. Sia $\mu^\star = \max_{i \in [K]} \mu^{(i)}$ il valore atteso del *reward* per l'*arm* migliore, e $\Delta_i = \mu^\star - \mu^{(i)} \geq 0$ sia il *gap* tra l'*arm* i e l'*arm* migliore. L'intero orizzonte temporale T viene suddiviso in M *batches* rappresentati da una *grid* $\mathcal{T} = \{t_1, \cdots, t_M\}$, con $1 \leq t_1 < t_2 < \cdots < t_M = T$, dove la *grid* appartine a una delle seguenti categorie:

1. *Grid statica*: la *grid* $\mathcal{T} = \{t_1, \cdots, t_M\}$ è prefissata prima della campionatura degli *arms*;
2. *Grid adattabile*: per $j \in [M]$, la *grid* può essere determinata dopo aver osservato i *rewards* fino al tempo t_{j-1} e usando un po' di casualità esterna.

Da notare che la *grid adattabile* è più pottente e pratica di quella statica e si ottiene *batch*

[5] William R. Thompson. «On the likelihood that one unknown probability exceeds another in view of the evidence of two samples». Inglese. In: *Biometrika* 25.3-4 (dic. 1933). Documento locale in pdf, pagine 285–294. ISSN: 0006-3444. DOI: 10.1093/biomet/25.3-4.285. eprint: https://academic.oup.com/biomet/article-pdf/25/3-4/285/513725/25-3-4-285.pdf.

[6] Herbert Robbins. «Some aspects of the sequential design of experiments». Inglese. In: *Bulletin of the American Mathematical Society* 58.5 (set. 1952), pagine 527–535. URL: https://projecteuclid.org:443/euclid.bams/1183517370.

[7] Aniket Kittur, Ed H. Chi e Bongwon Suh. «Crowdsourcing User Studies with Mechanical Turk». Inglese. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Documento locale in pdf. Florence, Italy: Association for Computing Machinery, apr. 2008, pagine 453–456. ISBN: 9781605580111. DOI: 10.1145/1357054.1357127.

[8] Dimitris Bertsimas e Adam J. Mersereau. «A Learning Approach for Interactive Marketing to a Customer Segment». Inglese. In: *Operations Research* 55.6 (nov. 2007). Documento locale in pdf, pagine 1120–1135. ISSN: 0030-364X. DOI: 10.1287/opre.1070.0427. URL: https://www.researchgate.net/publication/220244453.

[9] Stephen E. Chick e Noah Gans. «Economic Analysis of Simulation Selection Problems». Inglese. In: *Management Science* 55.3 (mar. 2009). Documento locale in pdf, pagine 421–437. DOI: 10.1287/mnsc.1080.0949. URL: https://www.researchgate.net/publication/227447666.

learning e *online learning* impostando $M = 1$ e $M = T$ rispettivamente.

Una *policy* di campionamento $\pi = (\pi_t)_{t=1}^{T}$ è una sequenza di variabili casuali $\pi_t \in [K]$ che indica quale *arm* tirare al tempo $t \in [T]$ dove per $t_{j-1} < t \leq t_j$, la *policy* π_t dipende solo dalle osservazioni fino al tempo t_{j-1}. In altre parole, la *policy* π_t dipende dalle osservazioni rigorosamente prima del *batch* attuale di t. L'obiettivo finale è di escogitare una *policy* di campionamento π per minimizzare il *regret* cumulativo atteso (o pseudo-*regret* o semplicemente *regret*), i.e., per minimizzare $\mathbb{E}[R_T(\pi)]$ dove

$$R_T(\pi) \triangleq \sum_{t=1}^{T} \left(\mu^* - \mu^{(\pi_t)}\right) = T\mu^* - \sum_{t=1}^{T} \mu^{(\pi_t)}.$$

Sia $\Pi_{M,T}$ l'insieme delle *policy* con M *batches* e l'orizzonte temporale T, l'obiettivo è di caratterizzare il seguente *minimax regret* e il *regret dipendente dal problema* sotto l'impostazione *batched*:

$$R^{\star}_{\min-\max}(K, M, T) \triangleq \inf_{\pi \in \Pi_{M,T}} \sup_{\{\mu(i)\}_{i=1}^{K} : \Delta_i \leq \sqrt{K}} \mathbb{E}[R_T(\pi)], \qquad \text{Espressione 1}$$

$$R^{\star}_{\text{pro-dep}}(K, M, T) \triangleq \inf_{\pi \in \Pi_{M,T}} \sup_{\Delta > 0} \Delta \cdot \sup_{\{\mu^{(i)}\}_{i=1}^{K} : \Delta_i \in \{0\} \cup [\Delta, \sqrt{K}]} \mathbb{E}[R_T(\pi)] \qquad \text{Espressione 2}$$

Notare che il *gap* tra gli *arms* può essere arbitrario nella definizione del *minimax regret*, mentre è presente un *lower bound* nei *gap* minimi nel *regret dipendente dal problema*. Il vincolo $\Delta_i \leq \sqrt{K}$ è una condizione tecnica in entrambi i scenari, che è più rilassato della consueta condizione $\Delta_i \in [0, 1]$. Queste grandezze sono motivate dal fatto che, quando $M = T$, gli *upper bounds* del *regret* per i *Multi-armed Bandit (MaB)* di solito prendono la forma[10][11][12][13]

$$\mathbb{E}\left[R_T\left(\pi^1\right)\right] \leq C\sqrt{KT},$$
$$\mathbb{E}\left[R_T\left(\pi^2\right)\right] \leq C \sum_{i \in [K] : \Delta_i > 0} \frac{\max\{1, \log(T\Delta_i^2)\}}{\Delta_i},$$

[10] Walter Vogel. «A Sequential Design for the Two Armed Bandit». Inglese. In: *Annals of Mathematical Statistics* 31.2 (giu. 1960). Documento locale in pdf, pagine 430–443. DOI: 10.1214/aoms/1177705906.

[11] Tze Leung Lai e Herbert Robbins. «Asymptotically efficient adaptive allocation rules». Inglese. In: *Advances in applied mathematics* 6.1 (mar. 1985). Documento locale in pdf, pagine 4–22. ISSN: 0196-8858. DOI: 10.1016/0196-8858(85)90002-8.

[12] Jean-Yves Audibert e Sébastien Bubeck. «Minimax policies for adversarial and stochastic bandits». Inglese. In: *COLT*. Documento locale in pdf. Montreal, Canada, giu. 2009, pagine 217–226. DOI: 10.1016/0196-8858(85)90002-8. URL: https://hal-enpc.archives-ouvertes.fr/hal-00834882.

[13] Sébastien Bubeck, Vianney Perchet e Philippe Rigollet. «Bounded regret in stochastic multi-armed bandits». Inglese. In: *Proceedings of the 26th Annual Conference on Learning Theory*. A cura di Shai Shalev-Shwartz e Ingo Steinwart. Volume 30. Proceedings of Machine Learning Research. Documento locale in pdf. Princeton, NJ, USA, giu. 2013, pagine 122–134. eprint: 1302.1611. URL: http://proceedings.mlr.press/v30/Bubeck13.html.

dove π^1, π^2 sono delle *policy* e $C > 0$ è una costante assoluta. Questi *bounds* sono anche stringenti, limitanti nel senso di *minimax*[11][12]. Di conseguenza, in un'impostazione adattabile (i.e., quando $M = T$), si hanno *regrets* ottimi $R^*_{\text{min}-\text{max}}(K,T,T) = \Theta(\sqrt{KT})$ e $R^\star_{\text{pro-dep}}(K,T,T) = \Theta(K \log T)$. L'obiettivo *target* è di scoprire la dipendenza di queste grandezze dal numero dei *batches* M.

Il primo risultato affronta gli *upper bounds* sul *minimax regret* e sul *regret dipendente dal problema*.

Teorema 1 (K, T, M - *policy*) *Per ogni $K \geq 2$, $T \geq 1$, $1 \leq M \leq T$, esistono due policy π^1 e π^2 in grid statiche (definita in sezione 4.3 - La BaSE Policy tale che se $\max_{i\in[K]} \Delta_i \leq \sqrt{K}$, allora abbiamo*

$$\mathbb{E}\left[R_T\left(\pi^1\right)\right] \leq \text{polylog}(K,T) \cdot \sqrt{K} T^{\frac{1}{2-2^{1-M}}},$$
$$\mathbb{E}\left[R_T\left(\pi^2\right)\right] \leq \text{polylog}(K,T) \cdot \frac{K T^{1/M}}{\min_{i\neq\star} \Delta_i},$$

dove $\text{polylog}(K,T)$ *nasconde fattori poli-logaritmici in* (K,T).

Il seguente corollario è immediato.

Corollario 1 (*M-batched K-armed Bandit* problem, $O(\log\log T)$) *Per il problema del MbKaB con orizzonte temporale T è sufficiente avere $M = O(\log\log T)$ batches per raggiungere l'ottimo minimax regret $\Theta(\sqrt{KT})$, e $M = O(\log T)$ per raggiungere l'ottimo regret dipendente dal problema $\Theta(K \log T)$, dove entrambi i regrets sono entro fattori logaritmici.*

Per i *lower bounds* del *regret*, si considerano la *grid statica* e la *grid adattabile* separatamente. Il successivo teorema presenta i *lower bounds* sotto qualsiasi *grid statica*.

Teorema 2 (*M-batched K-armed Bandit* problem, *grid statica*) *Per il problema del M-batched K-armed Bandit (MbKaB) con orizzonte temporale T e qualsiasi grid statica, il minimax regret e regret dipendente dal problema possono essere* lower bound*ed come segue*

$$R^\star_{\text{min}-\text{max}}(K,M,T) \geq c \cdot \sqrt{K} T^{\frac{1}{2-2^{1-M}}},$$
$$R^*_{pro\text{-}dep}(K,M,T) \geq c \cdot K T^{\frac{1}{M}},$$

dove $c > 0$ è una costante numerica indipendente da K, M, T.

Si osserva che per qualsiasi *grid statica*, i *lower bounds* nel Teorema 2 combaciano con quelli del Teorema 1 entro fattori poli-logaritmici. Per generiche *grid adattabili*, il seguente teorema mostra *regret lower bound* leggermente più deboli del Teorema 2.

Teorema 3 (*M-batched K-armed Bandit* problem, *grid adattabile*) *Per il problema del M-batched K-armed Bandit con orizzonte temporale T e qualsiasi grid adattabile, il minimax regret e regret dipendente dal problema possono essere lower bounded come segue*

$$R^{\star}_{\min-\max}(K,M,T) \geq cM^{-2}\cdot\sqrt{K}T^{\frac{1}{2-2^{1-M}}}$$
$$R^{\star}_{pro\text{-}dep}(K,M,T) \geq cM^{-2}\cdot KT^{\frac{1}{M}}$$

dove $c > 0$ è una costante numerica independente da K, M, T.

Rispetto al Teorema 2, i *lower bounds* nel Teorema 3 perdono un fattore polinomiale in M a causa di uno spazio più ampio di *policy*. Però, dato che il numero di *batches* di interesse è al più $O(\log T)$ (altrimenti per il Corollario 1 effettivamente si arriva al caso pienamente adattabile con $M = T$), questa penalità è al massimo poli-logaritmica in T. Conseguentemente, il Teorema 3 mostra che per qualsisasi *grid adattabile*, sebbene più potente, la sua *performance* è essenzialmente non migliore di quella della migliore *grid statica*. In particolare, si ha il seguente corollario:

Corollario 2 (*M-batched K-armed Bandit* problem, Omega(loglogT)) *Per il MbKaB problem con orizzonte temporale T con grid statiche o grid adattabili, è necessario avere $M = \Omega(\log\log T)$ batches per raggiungere l'ottimo minimax regret ottimale $\Theta(\sqrt{KT})$, e $M = \Omega(\log T/\log\log T)$ per raggiungere l'ottimo regret dipendente dal problema $\Theta(K\log T)$ dove entrambi i regrets ottimi sono entro fattori logaritmici.*

In sintesi, i risultati di cui sopra hanno caratterizzato completamente il *minimax regret* e il *regret dipendente dal problema* per problemi di *Batched Multi-armed Bandit (BMaB)*, entro fattori logaritmici. Una questione rilevante aperta è se il termine M^{-2} nel Teorema 3 possa essere rimosso usando argomenti più raffinati.

Capitolo 3

Studi correlati

Icona del Capitolo 3)[2]

3.1 Panoramica generale sui lavori correlati

Il problema del *Multi-armed Bandit (MaB)* fa parte di un'importante classe di problemi di ottimizzazione sequenziale che negli ultimi anni è stata ampiamente studiata in vari campi come la statistica, la ricerca operativa, l'ingegneria, l'informatica e l'economia[14].

Nello scenario completamente adattivo, l'analisi del *regret* per *stochastic bandits* si può trovare in numerosi studi[10][11][12][13][15][16][17].

[14] Sébastien Bubeck e Nicolò Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Inglese. Documento locale in pdf. 2012. DOI: 10.1561/2200000024.

[15] Apostolos N. Burnetas e Michael N. Katehakis. «Optimal Adaptive Policies for Markov Decision Processes». Inglese. In: *Mathematics of Operations Research* 22.1 (feb. 1997). Documento locale in pdf, pagine 222–255. ISSN: 0364765X, 15265471. DOI: 10.1287/moor.22.1.222. URL: http://www.jstor.org/stable/3690147.

[16] Peter Auer, Nicolò Cesa-Bianchi e Paul Fischer. «Finite-time Analysis of the Multiarmed Bandit Problem». Inglese. In: *Machine Learning* 47.2 (mag. 2002). Documento locale in pdf, pagine 235–256. ISSN: 1573-0565. DOI: 10.1023/A:1013689704352.

[17] Jean-Yves Audibert, Rémi Munos e Csaba Szepesvári. «Exploration–exploitation tradeoff using variance estimates in multi-armed bandits». Inglese. In: *Theoretical Computer Science* 410.19 (apr. 2009). Documento locale in pdf, pagine 1876–1902. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.01.016. URL: http://www.sciencedirect.com/science/article/pii/S030439750900067X.

Ulteriori approfondimenti[18][19][20][21].

C'è meno attenzione sull'impostazione batched con *rounds* di interazione limitati. L'impostazione batched viene studiata in «Online Learning with Switching Costs and Other Adaptive Adversaries» di Cesa-Bianchi, Dekel e Shamir[22] sotto il nome di *switching costs*, dove è mostrato che $O(\log \log T)$ *batches* sono sufficienti per raggiungere l'ottimo *minimax regret*. Per un piccolo numero di *batches* M, il problema del *Batched 2-armed Bandit (B2aB)* viene studiato in «Batched bandit problems» di Perchet, Rigollet, Chassang e Snowberg[4], dove i risultati dei Teorema 1 e Teorema 2 sono ottenuti per $K = 2$. Tuttavia, la generalizzazione al caso multi-*arm*ato non è semplice, e soprattutto, lo scenario pratico in cui la *grid* viene scelta in modo adattivo sulla base dei dati storici è escluso in «Batched bandit problems» di Perchet, Rigollet, Chassang e Snowberg[4].

Per il caso multi-*arm*ato, un diverso problema nel trovare i migliori k *arms* in un ambiente a impostazione *batched* è studiato in «Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls» di Jun, Jamieson, Nowak e Zhu[23] e in «Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons» di Agarwal,

[18] Jean-Yves Audibert e Sébastien Bubeck. «Regret Bounds and Minimax Policies under Partial Monitoring». Inglese. In: *Journal of Machine Learning Research* 11.94 (dic. 2010). Documento locale in pdf, pagine 2785–2836. ISSN: 1532-4435. URL: http://jmlr.org/papers/v11/audibert10a.html.

[19] Peter Auer e Ronald Ortner. «UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem». Inglese. In: *Periodica Mathematica Hungarica* 61.1 (set. 2010). Documento locale in pdf, pagine 55–65. ISSN: 1588-2829. DOI: 10.1007/s10998-010-3055-6.

[20] Aurélien Garivier e Olivier Cappé. «The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond». Inglese. In: *Proceedings of the 24th Annual Conference on Learning Theory*. A cura di Sham M. Kakade e Ulrike von Luxburg. Volume 19. Proceedings of Machine Learning Research. Documento locale in pdf. Budapest, Hungary, giu. 2011, pagine 359–376. URL: http://proceedings.mlr.press/v19/garivier11a.html.

[21] Vianney Perchet e Philippe Rigollet. «The multi-armed bandit problem with covariates». Inglese. In: *Annals of Statistics* 41.2 (apr. 2013). Documento locale in pdf, pagine 693–721. ISSN: 0090-5364. DOI: 10.1214/13-aos1101.

[22] Nicolò Cesa-Bianchi, Ofer Dekel e Ohad Shamir. «Online Learning with Switching Costs and Other Adaptive Adversaries». Inglese. In: *Advances in Neural Information Processing Systems 26*. A cura di C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani e K. Q. Weinberger. NIPS'13. Documento locale in pdf. Curran Associates, Inc., 2013, pagine 1160–1168. eprint: 1302.4387. URL: https://papers.nips.cc/paper/5151-online-learning-with-switching-costs-and-other-adaptive-adversaries.

[23] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak e Xiaojin Zhu. «Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls». Inglese. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. A cura di Arthur Gretton e Christian C. Robert. Volume 51. Proceedings of Machine Learning Research. Documento locale in pdf. Cadiz, Spain, mag. 2016, pagine 139–148. URL: http://proceedings.mlr.press/v51/jun16.html.

Agarwal, Assadi e Khanna[24], dove l'obiettivo è la pura esplorazione, e la dipendenza dall'errore dall'orizzonte temporale diminuisce in modo super-polinomiale. Ci si riferisce anche a «Minimax Bounds on Stochastic Batched Convex Optimization» di Duchi, Ruan e Yun[25] per un'impostazione simile con *bandits* convessi e migliore identificazione dell'*arm*. L'analisi del *regret* per *batched stochatic multi-armed bandits* rimane ancora poco esplorata. Si esaminano anche alcune pubblicazioni in letteratura sulla comuputazione generale con limitati *rounds* di adattamento, e in particolare sull'analisi dei *lower bounds*. Nell'informatica teorica, questo problema è stato studiato sotto il nome di algoritmi paralleli per determinati compiti (e.g., ordinamento e selezione), sia con risultati deterministici[26][27][28] che *noisy* (con rumori)[29][30][31].

Nell'ottimizzazione convessa (stocastica), i limiti teorici delle informazioni sono tipicamente derivati dal modello dell'oracolo in cui l'oracolo può essere interrogato in modo adattivo[32]

[24] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi e Sanjeev Khanna. «Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons». Inglese. In: *Proceedings of the 2017 Conference on Learning Theory*. A cura di Satyen Kale e Ohad Shamir. Volume 65. Proceedings of Machine Learning Research. Documento locale in pdf. Amsterdam, Netherlands, lug. 2017, pagine 39–75. URL: http://proceedings.mlr.press/v65/agarwal17c.html.

[25] John Duchi, Feng Ruan e Chulhee Yun. «Minimax Bounds on Stochastic Batched Convex Optimization». Inglese. In: *Proceedings of the 31st Conference On Learning Theory*. A cura di Sébastien Bubeck, Vianney Perchet e Philippe Rigollet. Volume 75. Proceedings of Machine Learning Research. Documento locale in pdf. Lug. 2018, pagine 3065–3162. URL: http://proceedings.mlr.press/v75/duchi18a.html.

[26] Leslie G. Valiant. «Parallelism in Comparison Problems». Inglese. In: *SIAM Journal on Computing* 4.3 (1975). Documento locale in pdf, pagine 348–355. DOI: 10.1137/0204030.

[27] Béla Bollobás e Andrew Thomason. «Parallel sorting». Inglese. In: *Discrete Applied Mathematics* 6.1 (1983). Documento locale in pdf, pagine 1–11. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(83)90095-1.

[28] Noga Alon e Yossi Azar. «Sorting, approximate sorting, and searching in rounds». Inglese. In: *SIAM Journal on Discrete Mathematics* 1.3 (ago. 1988). Documento locale in pdf, pagine 269–280. ISSN: 0895-4801. DOI: 10.1137/0401028.

[29] Uriel Feige, Prabhakar Raghavan e Eli Upfal. «Computing with Noisy Information». Inglese. In: *SIAM Journal on Computing* 23 (ott. 1994). Documento locale in pdf, pagine 1001–1018. ISSN: 0097-5397. DOI: 10.1137/S0097539791195877.

[30] Susan Davidson, Sanjeev Khanna, Tova Milo e Sudeepa Roy. «Top-k and Clustering with Noisy Comparisons». Inglese. In: *ACM Transactions on Database Systems* 39.4 (dic. 2014). Documento locale in pdf, pagine 1–39. ISSN: 0362-5915. DOI: 10.1145/2684066.

[31] Mark Braverman, Jieming Mao e S. Matthew Weinberg. «Parallel Algorithms for Select and Partition with Noisy Comparisons». Inglese. In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Documento locale in pdf. Association for Computing Machinery, 2016, pagine 851–862. ISBN: 9781450341325. DOI: 10.1145/2897518.2897642.

[32] Arkadii Semenovich Nemirovsky e David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. Inglese. Wiley-Interscience series in discrete mathematics. John Wiley & Sons, 1983. 388 pagine. ISBN: 9780471103455. URL: http://www.worldcat.org/oclc/252233116.

(Originale in lingua russa:[33])[34][35][25]. Tuttavia, in studi precedenti, di solito si ottimizzava in ogni *step* la distribuzione del campionamento su una dimensione fissa del campione, mentre è più difficile dimostrare *lower bounds* per le *policies* che possono anche determinare la dimensione del campione. C'è un'eccezione («Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons» di Agarwal, Agarwal, Assadi e Khanna[24]), la cui dimostrazione si basa su una complicata decomposizione di distribuzioni quasi uniformi. Quindi, la tecnica per dimostrare il Teorema 3 si aspetta sia un contributo aggiunto a queste pubblicazioni.

3.2 Elenco degli studi citati nel *paper* originale

Lista intera degli studi citati nel *paper* originale, con *citing*, relativa descrizione e documento allegato:

«Batched bandit problems» Perchet, Rigollet, Chassang e Snowberg (2016)[4] Studia il *regret* raggiungibile per i *stochastic bandits* sotto la *policy* di suddividere i *trials* in un piccolo numero di *batches*. Propone una semplice *policy* e mostra che un numero molto piccolo di *batches* fornisce *bounds* vicini al *minimax regrets* ottimale. Inoltre fornisce delle *policies* ottimali con basso costo di *switching* per *stochastic bandits*.
Documento locale in pdf

«On the likelihood that one unknown probability exceeds another in view of the evidence of two samples» Thompson (1933)[5] Studio per la scelta di azioni che affrontano il dilemma *exploration-exploitation*) nel problema dei *Multi-armed Bandit*. Consiste nella scelta dell'azione che massimizza il *reward* atteso rispetto a una convinzione disegnata casualmente.
Documento locale in pdf

[33] Arkadij Semenovič Nemirovskij e David Borisovič Judin. *Ložnost' zadač i èffektivnost' metodov optimizacii*. Russo. Nauka, 1979. 384 pagine.

[34] Alekh Agarwal, Martin J. Wainwright, Peter L. Bartlett e Pradeep K. Ravikumar. «Information-theoretic lower bounds on the oracle complexity of convex optimization». Inglese. In: *Advances in Neural Information Processing Systems 22*. A cura di Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams e A. Culotta. Documento locale in pdf. Curran Associates, Inc., 2009, pagine 1–9. eprint: 1009.0571. URL: https://papers.nips.cc/paper/3689-information-theoretic-lower-bounds-on-the-oracle-complexity-of-convex-optimization.

[35] Ohad Shamir. «On the Complexity of Bandit and Derivative-Free Stochastic Convex Optimization». Inglese. In: *Proceedings of the 26th Annual Conference on Learning Theory*. A cura di Shai Shalev-Shwartz e Ingo Steinwart. Volume 30. Proceedings of Machine Learning Research. Documento locale in pdf. Princeton, NJ, USA, giu. 2013, pagine 3–24. URL: http://proceedings.mlr.press/v30/Shamir13.html.

«Some aspects of the sequential design of experiments» Robbins (1952)[6] Si propone la *Hedge* come uno schema adattivo, che guida la mano del giocatore in un gioco *Multi-armed Bandit* con informazioni complete Le applicazioni di questo gioco esistono nella selezione del percorso di rete, nella distribuzione del carico e nel *network interdiction*. Si esegue un'analisi del caso peggiore dell'algoritmo *Hedge* utilizzando un avversario, che selezionerà costantemente le penalità in modo da massimizzare la perdita del giocatore, supponendo che il budget di penalità dell'avversario sia limitato. Si esplorano ulteriormente le prestazioni delle penalità binarie e si dimostra che la strategia binaria ottimale per l'avversario è prendere decisioni avide.

«Crowdsourcing User Studies with Mechanical Turk» Kittur, Chi e Suh (2008)[7] Si esamina l'utilità di un mercato di micro-attività per la raccolta delle misurazioni degli utenti e si discute di considerazioni di progettazione per lo sviluppo di attività di valutazione di micro utenti remoti.
Documento locale in pdf

«A Learning Approach for Interactive Marketing to a Customer Segment» Bertsimas e Mersereau (2007)[8] Quando un operatore di marketing in un ambiente interattivo decide quali messaggi inviare ai clienti, può inviare messaggi attualmente ritenuti i più promettenti (*exploitation*) oppure utilizzare messaggi a scopo di raccolta di informazione (*exploration*). Partendo dal presupposto che i clienti siano già raggruppati in segmenti omogenei, si considera l'efficacia dell'apprendimento adattivo all'interno di un segmento di clientela. Si presenta una formulazione bayesiana del problema in cui vengono prese decisioni simultaneamente per *batches* di clienti, sebbene le decisioni possano variare all'interno di un *batch*. Questo estende il classico problema del *Multi-armed Bandit* per campionare uno a uno da una serie di popolazioni di *rewards*. Il metodo di soluzione proposto include un approccio di programmazione dinamica approssimativa basato sulla decomposizione Lagrangiana e un metodo euristico basato su una nota approssimazione asintotica della soluzione di *Multi-armed Bandit*. I risultati computazionali mostrano chiaramente che i metodi proposti sono migliori degli approcci che ignorano gli effetti dell'*information gain*.
Documento locale in pdf

«Economic Analysis of Simulation Selection Problems» Chick e Gans (2009)[9] Il *paper* riassume un nuovo approccio recentemente proposto per i problemi di *ranking* e *selection*, che massimizza il Net Present Value (NPV) previsto dalle decisioni prese usando simulazioni stocastiche o simulazioni *discrete-event*. La formulazione proposta presuppone che esistano le condizioni per simulare un numero fisso di progetti alternativi, e si propone il problema come una versione "arrestabile" del problema del *bandit* bayesiano. Sotto condizioni relativamente generali, un indice di Gittins

può essere usato per indicare quale sistema simulare o implementare. Si fornisce un'approssimazione asintotica per l'indice che è appropriata quando gli *output* della simulazione normalmente distribuiti con note ma potenzialmente differenti varianze per i diversi sistemi. I costi includevano i costi variabili della simulazione e attualizzazione dovuti al tempo di analisi della simulazione.
Documento locale in pdf

«A Sequential Design for the Two Armed Bandit» Vogel (1960)[10] Propone una classe di strategie per far massimizzare la somma degli *outcomes* di una sequenza di decisioni di uno sperimentatore nello scegliere i futuri step assumendo *a priori* o non le distribuzioni dei valori di p e q, ovvero degli *outcomes* dei due esperimenti.
Documento locale in pdf

«Asymptotically efficient adaptive allocation rules» Lai e Robbins (1985)[11] Descrive metodi generali per costruire *adaptive allocation rules* che raggiungono il limite inferiore asintotico per il *regret*. Al fine di raggiungere l'efficienza asintotica, si prendono *sample* dalla popolazione con la media attesa più grande, a condizione che sia stata campionata abbastanza da ogni popolazione per essere ragionevolmente *confident*.
Documento locale in pdf

«Minimax policies for adversarial and stochastic bandits» Audibert e Bubeck (2009)[12] Colma un vuoto nella caratterizzazione della *minimax rate* per il problema del *Multi-armed Bandit*. Propone una nuova famiglia di algoritmi randomizzati basato su una normalizzazione implicita, nonché una nuova analisi. Si considera anche il caso stocastico, e si dimostra che una modifica appropriata della *policy* UCB1 raggiunge il *distribution-free optimal rate* pur avendo ancora un numero di giocate con *distribution-dependent rate* logaritmico.
Documento locale in pdf

***Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems* | Bubeck e Cesa-Bianchi (2012)**[14] In questo paper, il *focus* è posto su due casi estremi in cui l'analisi del *regret* per il problema del *Multi-armed Bandit* è particolarmente semplice e elegante: *payoffs* e *adversarial payoffs* i.i.d.. Oltre alle impostazioni di base di molte azioni finite, si analizzano anche alcune delle più importanti varianti ed estensioni, come *contextual* bandit *model*, *linear* e *nonlinear* bandits.
Documento locale in pdf

«Optimal Adaptive Policies for Markov Decision Processes» Burnetas e Katehakis (1997)[15] Si considera il problema del controllo adattivo per i *Markov Decision Processes*. Si da la forma

esplicita per una classe di *policies* adattabili che possiedono proprietà di tasso di aumento ottimali per il *reward* totale previsto nell'orizzonte finito, sotto ipotesi sufficienti di spazio-azione finiti e irriducibilità della legge di transizione. Una caratteristica principale delle *policies* proposte è che la scelta delle azioni, in ogni stato e periodo di tempo, si basa su indici che sono inflazioni del lato destro delle equazioni di ottimalità del *reward* medio atteso.

Documento locale in pdf

«Exploration–exploitation tradeoff using variance estimates in multi-armed bandits» Audibert, Munos e Szepesvári (2009)[17] Questo *paper* considera una variante dell'algoritmo di base (su limiti di confidenza superiori per bilanciare *exploration* e *exploitation*) per il problema stocastico del *Multi-armed Bandit* che tiene conto della varianza empirica dei diversi *arms*. Si forna la prima analisi del *regret* atteso per tali algoritmi. I risultati mostrano che l'algoritmo che utilizza le stime di varianza ha un grande vantaggio rispetto alle alternative che non utilizzano tali stime a condizione che le variazioni dei *rewards* degli *arms* non ottimali siano bassi. Si dimostra anche che il *regret* si concentra solo a un tasso polinomiale. Questo vale per tutti gli algoritmi basati sul limite di confidenza superiore e per tutti i *bandits problems* eccetto alcuni casi. Quindi, sebbene gli algoritmi dei *bandits* legati alla fiducia superiore ottengano tassi di *regret* atteso logaritmici, potrebbero non essere adatti per un decisore avverso al rischio. Si illustrano alcuni dei risultati mediante simulazioni al computer.

Documento locale in pdf

«Regret Bounds and Minimax Policies under Partial Monitoring» Audibert e Bubeck (2010)[18] Questo lavoro si occupa di quattro impostazioni di previsione classiche, vale a dire informazioni complete, *bandit*, *label efficient* e *bandit label efficient* oltre a quattro diverse nozioni di *regret*: pseudo-*regret*, atteso *regret*, *regret* ad alta probabilità e monitoraggio del miglior *regret* esperto. Si presenta un nuovo *forecaster*, *Implicitly Normalized Forecaster (INF)* basato su una funzione arbitraria ψ per la quale si propone un'analisi unificata del suo pseudo-*regret* nei quattro giochi che si considerano. Infine, si considera il gioco del *stochastic bandit* e si dimostra che una modifica appropriata della confidenza superiore nella *policies* UCB1 raggiunge la *distribution-free optimal rate* pur avendo un *distribution-dependent rate* logaritmica nel numero di giocate.

Documento locale in pdf

«UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem» Auer e Ortner (2010)[19] Nel problema stocastico del *Multi-armed Bandit* si considera una mo-

difica dell'algoritmo UCB. Per questo modificato algoritmo si ha un migliore limite al *regret* con rispetto al *reward* ottimale.
Documento locale in pdf

«The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond» Garivier e Cappé (2011)[20] Questo *paper* presenta un'analisi a tempo finito dell'algoritmo KL-UCB, una *horizon-free index policy* per problemi di *stochastic bandit*. Si dimostrano due risultati distinti: primo, per *reward* limitati arbitrari, l'algoritmo KL-UCB soddisfa un *bound* del *regret* uniformemente migliore rispetto a UCB e alle sue varianti; secondo, nel caso speciale dei *rewards* di Bernoulli, esso raggiunge il limite inferiore di Lai e Robbins. Inoltre, si mostra che semplici adattamenti dell'algoritmo KL-UCB sono ottimali anche per classi specifiche di (possibilmente illimitate) *rewards*, comprese quelle generate da famiglie esponenziali di distribuzioni. Uno studio numerico a larga scala che confronta KL-UCB con i suoi principali concorrenti (UCB, MOSS, UCB-Tuned, UCB-V, DMED) mostra che KL-UCB è straordinariamente efficiente e stabile, anche in brevi orizzonti temporali. KL-UCB è anche l'unico metodo che funziona sempre meglio della *policy* di base UCB.
Documento locale in pdf

«Bounded regret in stochastic multi-armed bandits» Bubeck, Perchet e Rigollet (2013)[13] Si studia il problema stocastico del *Multi-armed Bandit* quando si conosce il valore $\mu^{(*)}$ di un *arm* ottimale, oltre che un limite inferiore positivo legato al minimo *gap* positivo Δ. Si propone una nuova *policy* randomizzata che, in questa impostazione, ottiene un *regret* uniformemente limitato nel tempo. Si dimostrano anche diversi limiti inferiori, che mostrano in particolare che non è possibile limitare il *regret* se si conosce solo Δ e un *regret* limitato all'ordine $1/\Delta$ non è possibile se si conosce solo $\mu^{(*)}$.
Documento locale in pdf

«Online Learning with Switching Costs and Other Adaptive Adversaries» Cesa-Bianchi, Dekel e Shamir (2013)[22] Si studia il potere di diversi tipi di avversari adattivi (*nonoblivious*) nell'impostazione della predizione con *expert advice*, sia sotto informazione completa che dì *bandit feedback*. Si misurano le prestazioni del giocatore usando una nuova nozione di *regret*, noto anche come *policy regret*, che cattura meglio l'adattamento dell'avversario al comportamento del giocatore. In un ambiente in cui le perdite possono andare alla deriva, caratterizzare, in modo quasi completo, il potere degli avversari adattivi con memorie limitate e costi di *switching*.
Documento locale in pdf

«Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls» Jun, Jamieson, Nowak e Zhu (2016)[23] Si presenta il problema *Multi-armed Bandit (MaB)* in cui gli *arms* devono essere campionati in *batches*, piuttosto che uno alla volta. Questo è motivato da applicazioni nei *social media* e sperimentazione biologica dove vincoli sui *batch* sorgono naturalmente. Il *paper* sviluppa e analizza anche algoritmi per *Batched Multi-armed Bandit (BMaB)* e identificazione del *top arm*, sia per *confidence* fissa che fissa impostazione del *budget*. Si dimostra il nuovo algoritmi per BMaB con simulazioni e in due interessanti applicazioni del mondo reale: (i) esperimenti con array di micropozzetti per l'identificazione di geni importanti nella replicazione di virus, e (ii) trovare gli utenti più attivi in Twitter su un argomento specifico.
Documento locale in pdf

«Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons» Agarwal, Agarwal, Assadi e Khanna (2017)[24] Si studia la relazione tra complessità di *query* e adattabilità nell'identificare le k monete più *biased* tra una serie di n monete con *bias* sconosciuti. Questo problema è una comune astrazione di molti problemi ben studiati, incluso il problema di identificare i migliori k *arms* di un stocastico *Multi-armed Bandit*, e il problema della classifica *top-k* da confronti di coppie. Si considera la domanda su quanta adattività è necessaria per capire la complessità ottimale peggiore di *query* per identificare le k monete più *biased*. L'algoritmo proposto raggiunge la complessità ottimale di *query* in al massimo $\log^* n$ *rounds*, il che implica che su qualsiasi input realistico, 5 *round* paralleli di esplorazione sono sufficienti per ottenere la complessità peggiore ottimale del campionamente.
Documento locale in pdf

«Parallelism in Comparison Problems» Valiant (1975)[26] Lo studio indaga la complessità nel caso peggiore di algoritmi per computer multiprocessore con confronti binari durante operazioni di base. Si dimostra che per i problemi di ricerca del massimo, l'ordinamento e l'unione di una coppia di liste ordinate, se n è la dimensione del *set* di *input*, delle accelerazioni di almeno $O(k/\log\log k)$ possono essere raggiunte rispetto a operazioni di confronto (k è il numero di processori). L'algoritmo per trovare il massimo si dimostra ottimale per tutti i valori di k e n.
Documento locale in pdf

«Parallel sorting» Bollobás e Thomason (1983)[27] Si dimostra che se n è sufficientemente grande, allora c'è un grafico G di ordine n con $\lfloor n^{3/2}\log n\rfloor$ *edges* tale che la chiusura transitiva di ogni orientamento aciclico di G abbia almeno $\binom{n}{2} - n^{3/2}\log n$ *edges*. Una conseguenza di ciò è che con $\lfloor n^{3/2}\log n\rfloor$ processori paralleli possono essere ordinati n oggetti in due intervalli di tempo.

Questo migliora notevolmente alcuni risultati di Häggkvist e Hell. Si dimostrano affermazione simili sull'ordinamento con solo d-*step* implicazioni.
Documento locale in pdf

«Sorting, approximate sorting, and searching in rounds» Alon e Azar (1988)[28] Questo studio propone: (a) Miglioramenti dei limiti noti precedentemente che separano gli algoritmi deterministici da quelli randomizzati. (b) Un modo per separare il problema di trovare la mediana da quello di trovare il minimo. (c) Una soluzione per il problema di Rabin, mostrando che "l'ordinamento approssimativo" in un round richiede asintoticamente più di $c \cdot n \log n$ confronti, per ogni costante c, e può essere fatto in $O(n \log n \log \log n)$ confronti.
Documento locale in pdf

«Computing with Noisy Information» Feige, Raghavan e Upfal (1994)[29] Questo documento studia la profondità dei alberi decisionali "con rumore" (*noisy*) decisionali in cui ogni nodo con qualche probabilità costante dà la risposta sbagliata. Nel modello dell'albero decisionale Booleano con rumore, vengono indicati limiti stretti sul numero di *queries* alle variabili *input* necessarie per calcolare le funzioni *threshold*, la funzione di parità e le funzioni simmetriche. Nel modello dell'albero di confronto con rumore, vengono indicati limiti stretti sul numero di confronti rumorosi per la ricerca, l'ordinamento, la selezione e il *merging*. L'articolo studia anche la selezione parallela e l'ordinamento con confronti rumorosi, fornendo limiti stretti per diversi problemi.
Documento locale in pdf

«Top-k and Clustering with Noisy Comparisons» Davidson, Khanna, Milo e Roy (2014)[30] In questa pubblicazione vengono studiano i problemi di max/top-k e *clustering* quando le operazioni di confronto possono essere eseguite da oracoli la cui risposta può essere errata. Si propongono algoritmi efficienti che sono garantiscono risultati corretti con alta probabilità, si analizza il costo di questi algoritmi in termini di numero totale di confronti (ovvero, utilizzando un modello a costo fisso), e si mostra che sono essenzialmente i migliori possibili. Si mostra anche che sono necessari meno confronti quando i valori e tipi sono correlati, o quando il modello di errore è uno in cui l'errore diminuisce con l'aumento della distanza tra i due elementi nell'ordine ordinato. Infine, si esamina un'altra importante classe di funzioni di costo, funzioni concave, che bilanciano il numero di round di interazione con l'oracolo con il numero di domande poste all'oracolo. I risultati di questo articolo rappresentano un primo passo importante nel fornire una base formale per max/top-k e *query* di *clustering* in applicazioni di *crowdsourcing*, ovvero quando l'oracolo viene implementato usando la *crowd*.

Documento locale in pdf

«Parallel Algorithms for Select and Partition with Noisy Comparisons» Braverman, Mao e Weinberg (2016)[31] L'articolo considera il problema di trovare il k-esimo elemento più alto in un insieme totalmente ordinato di n elementi (*SELECT*) e partizionare un insieme ordinato nella parte *top* k e *bottom* $n-k$ elementi (*PARTITION*) usando confronti a coppie. Motivati da impostazioni come la *peer grading* o il *crowdsourcing*, dove più cicli di interazione sono costosi e confronti a interrogazione possono essere incompatibili con la verità fondamentale, si valutano algoritmi basati sia sul loro tempo di esecuzione totale e il numero di round interattivi in tre confronti modelli: senza rumori, silenzioso (*noiseless*) dove i confronti sono corretti, a cancellazione (*erasure* dove i confronti vengono cancellati con probabilità $1-\gamma$), e con rumore (*noisy* dove i confronti sono corretti con probabilità $1/2+\gamma/2$ o altrimenti non corretti. I risultati nel modello silenzioso, che è abbastanza ben studiato nella letteratura TCS sugli algoritmi paralleli, sono nuovi.
Documento locale in pdf

***Problem complexity and method efficiency in optimization* | Nemirovsky e Yudin (1983)**[32]. Originale in lingua russa: ***Ložnost' zadač i èffektivnost' metodov optimizacii* | Nemirovskij e Judin (1979)**[33] In questo libro gli studiosi studiano algoritmi per minimizzare una funzione convessa soggetta a vincoli convessi. La quantità di lavoro svolto dall'algoritmo viene identificata con il numero di punti in cui le funzioni devono essere valutate per garantire un determinato grado di precisione. L'analisi è del tipo *worst-case*. Tuttavia, viene discussa la quantità prevista di lavoro per algoritmi che comportano una ricerca casuale. Vengono approfonditi due algoritmi: il metodo dei centri di gravità (MCG) e il metodo della discesa speculare (MMD). Vengono stabiliti risultati con limite inferiore per mostrare che (MCG) è quasi-ottimale (o sub-ottimale) per problemi su domini arbitrari, mentre (MMD) è sub-ottimale per problemi con molte variabili, un dominio uniforme e un limite alla precisione necessaria. Il libro offre preziose informazioni sull'algoritmo ellissoide, che corrisponde al loro (MMCG). I problemi di programmazione lineare non vengono discussi esplicitamente.

«Information-theoretic lower bounds on the oracle complexity of convex optimization» Agarwal, Wainwright, Bartlett e Ravikumar (2009)[34] In questo articolo, si studia la complessità dell'ottimizzazione convessa stocastica in un modello di calcolo oracolo. Si ottengono miglioramenti su risultati noti e stime ristrette di complessità *minimax* per varie classi di funzioni. Inoltre si discuttono le implicazioni di questi risultati per la comprensione della complessità intrinseca di problemi di apprendimento e di stima su larga scala.

Documento locale in pdf

«On the Complexity of Bandit and Derivative-Free Stochastic Convex Optimization» Shamir (2013)[35] In questo documento, si indaga l'errore/*regret* ottenibile nelle impostazioni *bandit* e *derivative-free*, come una funzione della dimensione d e del numero disponibile di *query* T. Si fornisce una precisa caratterizzazione delle prestazioni ottenibili per funzioni fortemente convesse e fluide, che implica anche un limite inferiore non banale per problemi più generali. Inoltre, si dimostra che sia nell'impostazione *bandit* che in quella *derivative-free*, il numero richiesto di *query* deve crescere almeno quadraticamente con la dimensione. Infine, si dimostra che sulla classe naturale di funzioni quadratiche, è possibile ottenere un tasso di errore $O(1/T)$ "rapido" in termini di T, sotto ipotesi lievi, anche senza ricorrere ai gradienti. Secondo gli autori, questo è il primo *rate* di questo tipo in un ambiente stocastico privo di derivati (*derivative-free*), e vale nonostante i precedenti risultati sembrano implicare il contrario.
Documento locale in pdf

«Minimax Bounds on Stochastic Batched Convex Optimization» Duchi, Ruan e Yun (2018)[25] Nel *paper* si studia il problema di ottimizzazione convessa in *batch* stocastico, in cui si usano molte osservazioni parallele per ottimizzare una funzione convessa dati *rounds* limitati di interazione. In ciascuno dei *rounds* M, un algoritmo può richiedere informazioni in n punti e dopo aver emesso tutte le n *query*, riceve una funzione con rumore (*noisy*) *biased* e/o valutazioni del (sub)gradiente negli n punti. Dopo M tali *rounds*, l'algoritmo deve generare uno stimatore. Si forniscono limiti inferiori e superiori per le prestazioni di tali algoritmi di ottimizzazione convessa in *batch* in impostazione di ordine zero e del primo ordine per funzioni convesse e lisce fortemente convesse di Lipschitz. I *rates* di convergenza dello studio (quasi) raggiungono pienamente il *rate* sequenziale una volta che $M = O(dloglogn)$, dove d è la dimensione del problema, però i *rates* possono "degradare" esponenzialmente all'aumentare della dimensione d, a differenza delle impostazioni completamente sequenziali.
Documento locale in pdf

«Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems» Even-Dar, Mannor e Mansour (2006)[36] Lo studio incorpora intervalli di confidenza statistica sia nel *Multi-armed Bandit* che nei problemi di *reinforcement*

[36] Eyal Even-Dar, Shie Mannor e Yishay Mansour. «Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems». Inglese. In: *Journal of Machine Learning Research* 7 (dic. 2006). Documento locale in pdf, pagine 1079–1105. ISSN: 1532-4435.

learning. Nel problema del *bandit* si mostra che dati *n arms*, è sufficiente tirare gli *arms* per un totale di $O(n/\varepsilon^2)\log(1/\delta)$ volte per trovare un *arm* ε-ottimale con probabilità di almeno $1-\delta$. Questo limite corrisponde al limite inferiore di Mannor e Tsitsiklis fino alle costanti. Si sono anche ideate procedure di eliminazione dell'azione negli algoritmi di apprendimento del rinforzo. Si descrive un quadro che si basa sull'apprendimento dell'intervallo di confidenza attorno alla funzione valore o alla funzione Q e eliminando azioni non ottimali. Si fornisce poi un metodo di eliminazione con varianti *model-based* e *model-free*. Inoltre si ottengono condizioni di arresto garantendo che la *learning policy* è approssimativamente ottimale con alta probabilità. Le simulazioni dimostrano una notevole accelerazione e maggiore robustezza rispetto al *greedy Q-learning*.
Documento locale in pdf

«The multi-armed bandit problem with covariates» Perchet e Rigollet (2013)[21] Si considera un problema di *Multi-armed Bandit* in un ambiente in cui ogni *arm* produce una realizzazione di *rewards* con rumore che dipende da una covariate casuale. A differenza del tradizionale problema del *Multi-armed Bandit* statico, questa impostazione consente di modificare in modo dinamico i *rewards* che descrivono meglio applicazioni in cui è disponibile *side information*. Si adotta un modello nonparametrico dove le *rewards* attese sono *smooth functions* della covariata e dove la *hardness* del problema viene catturata da un parametro *margin*. Per massimizzare la *reward* cumulativa attesa, si introduce una *policy* chiamata *Adaptively Binned Successive Elimination (ABSE)* che in modo adattivo decompone il problema globale in problemi di *bandit* statico opportunamente "localizzati". Questa *policy* costruisce un partizionamento adattivo usando una variante della *policy Successive Elimination (SE)*. I risultati includono limiti di *regret* più netti per la *policy SE* in un problema di *bandit* statico e limiti del *minimax regret* ottimale per la *policy ABSE* nel problema dinamico.
Documento locale in pdf

«Regret Bounds for Batched Bandits» Esfandiari, Karbasi, Mehrabian e Mirrokni (2019)[37] Si presentano algoritmi semplici ed efficienti per il *Batched Multi-armed Bandit* stocastico e il *Batched Linear Bandit* stocastico. Si dimostrano i limiti per i loro *regrets* attesi che migliorano ai migliori limiti di *regret* noti per qualsiasi numero di *batches*. In particolare, gli algoritmi in entrambe le impostazioni raggiungono i *regrets* ottimali attesi utilizzando solo un numero logaritmico di *batches*. Inoltre si studia per la prima volta anche il problema del *Batched Multi-*

[37] Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian e Vahab S. Mirrokni. «Regret Bounds for Batched Bandits». Inglese. In: *arXiv: Data Structures and Algorithms* (2019). Documento locale in pdf. URL: http://arxiv.org/abs/1910.04959v2.

armed Bandit avversario e si trova il *regret* ottimale, fino ai fattori logaritmici, di qualsiasi algoritmo con dimensioni di *batch* predeterminate.
Documento locale in pdf

***Elements of Information Theory* Cover e Thomas (2006)**[38] Libro di testo sulla teoria dell'informazione. Tratta gli argomenti in dettaglio, tra cui entropia, compressione dei dati, capacità del canale, distorsione della frequenza, teoria dell'informazione sulla rete e verifica delle ipotesi. Gli autori forniscono ai lettori una solida conoscenza della teoria e delle applicazioni sottostanti. I problemi e un riassunto telegrafico alla fine di ogni capitolo aiutano ulteriormente i lettori. Le note storiche che seguono ogni capitolo riassumono i punti principali.

Documento locale in pdf

«Finite-time Analysis of the Multiarmed Bandit Problem» | Auer, Cesa-Bianchi e Fischer (2002)[16] In questo lavoro si mostra che il *regret* logaritmico ottimale è raggiungibile uniformemente nel tempo, con *policies* semplici ed efficienti, e per tutte le distribuzioni di *rewards* con supporto limitato.
Documento locale in pdf

[38] T. M. Cover e J. A. Thomas. *Elements of Information Theory*. Inglese. Wiley Series in Telecommunications and Signal Processing. Documento locale in pdf. Wiley, 2006. 776 pagine. ISBN: 9780471748816. URL: https://books.google.it/books?id=0QuawYmc2pIC.

Capitolo 4

Descrizione

Icona del Capitolo 4[2]

4.1 Organizzazione dello studio

Lo studio è organizzato come segue: Nella sezione 4.3 - La BaSE *Policy* si introduce la politica BaSE per problemi di *Multi-armed Bandit (MaB)* generali, e si dimostra che raggiunge i limiti superiori nel Teorema 1 con due *grid* specifiche. La sezione 4.4 - Il *Lower bound* presenta le dimostrazioni dei *lower bounds* sia per il *minimax regret* sia per il *regret dipendente dal problema*, e in sottosezione 4.4.1 - La *grid statica* si trattano le *grid statiche* e in sottosezione 4.4.2 - La *grid adattabile* le *grid adattabili*. I risultati sperimentali sono presentati in sottosezione 5.1.3 - Risultati e osservazioni, sottosezione 5.2.3 - Risultati, confronti e osservazioni e in sottosezione 5.3.4 - Risultati e osservazioni. I lemmi ausiliari e le dimostrazioni dei lemmi principali sono mostrati nei materiali supplementari del *paper* oggetto di studio.

4.2 La notazione

Per un numero intero positivo n, sia $[n] \triangleq \{1, \cdots, n\}$. Per ogni insieme finito A, sia $|A|$ la sua cardinalità. Si adottano le notazioni asintotiche standard: per due sequenze non negative $\{a_n\}$ e $\{b_n\}$ sia $a_n = O(b_n)$ se e solo se (sse) $\limsup_{n\to\infty} a_n/b_n < \infty$, $a_n = \Omega(b_n)$ sse $b_n = O(a_n)$ e $a_n = \Theta(b_n)$ sse $a_n = O(b_n)$ e $b_n = O(a_n)$. Per misure di probabilità P e Q, sia $P \otimes Q$ la misura del prodotto con i margini P e Q. Se le misure P e Q sono definite sullo stesso spazio di probabilità, si indica con $\mathrm{TV}(P, Q) = \frac{1}{2}\int |dP - dQ|$ e $D_{\mathrm{KL}}(P\|Q) = \int dP \log \frac{dP}{dQ}$ la distanza di variazione totale e le divergenze Kullback-Leibler (KL) tra P e Q, rispettivamente.

4.3 La BaSE *Policy*

In questa sezione, si propone la *Batched Successive Elimination (BaSE) policy* per il problema dei *Batched Multi-armed Bandit (BMaB)* basato sull'eliminazione successiva, nonché due scelte delle *grid statiche* per dimostrare il Teorema 1.

4.3.1 Descrizione della *policy*

La *policy* che raggiunge i *regrets* ottimali è essenzialmente adattata da *Successive Elimination (SE)*. La versione originale di SE introdotta in «Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems» di Even-Dar, Mannor e Mansour[36] e «The multi-armed bandit problem with covariates» di Perchet e Rigollet[21] mostra che nel caso $M = T$, SE raggiunge sia il *minimax* che il *rate* dipendente dal problema. Nel *paper* si presenta una versione in *batch* di SE chiamata *Batched Successive Elimination (BaSE)* per gestire il caso generale $M \leq T$. Data una *grid* pre-specificata $\mathcal{T} = \{t_1, \cdots, t_M\}$, l'idea della politica BaSE è semplicemente quella di esplorare nel primo $M - 1$ batch e quindi impegnarsi all'*arm* migliore nell'ultimo *batch*. Alla fine di ogni *batch* di esplorazione, si rimuovono gli *arms* che probabilmente sono non buoni in base alle osservazioni passate. In particolare, sia $\mathcal{A} \subseteq \mathcal{I}$ la notazione degli *arms* attivi che sono candidate per l'*arm* ottimale, dove inizializziamo $\mathcal{A} = \mathcal{I}$ e rilasciamo sequenzialmente gli *arms* che sono "significativamente" peggiori di quelli "migliori". Per i primi $M - 1$ *batches*, tiriamo tutti gli *arms* attive per lo stesso numero di volte (trascurando il problema dell'arrotondamento [1]ed eliminando alcuni *arms* da $\mathcal{A}$ alla fine di ogni *batch*. Per l'ultimo *batch*, ci si impegna all'*arm* $\mathcal{A}$ con il massimo *reward* medio.

Prima di specificare l'algoritmo proprio, si introducono alcune notazioni. Sia

$$\bar{Y}^i(t) = \frac{1}{|\{s \leq t : \text{arm } i \text{ is pulled at time } s\}|} \sum_{s=1}^{t} Y_s \mathbb{1}\{\text{arm } i \text{ is pulled at time } s\}$$

denota i *rewards* medi dell'*arm* i fino al tempo t, e $\gamma > 0$ è un parametro di *tuning* associato al *Upper Confidence Bound (UCB)*. L'algoritmo è descritto in dettaglio nel Frammento di codice 4.1.

Si noti che l'algoritmo BaSE non è completamente specificato a meno che non venga determinata la *grid* $\mathcal{T}$. Qui si forniscono due scelte di *grid statiche*, che sono simili a quanto descritto in

[1] Qui potrebbero esserci alcuni problemi di arrotondamento e alcuni *arms* potrebbero essere tirati una volta più di altri. In questo caso, la tirata aggiuntiva non verrà conteggiata ai fini del calcolo del *reward* medio $\bar{Y}^i(t)$, che garantisce che tutti gli *arms* attivi vengano valutate usando lo stesso numero di tirate alla fine di ogni *batch*. Si noti che in questo modo, il numero di tirate per ciascun *arms* è sottovalutato al massimo dalla metà, quindi l'analisi del *regret* nel Teorema 4 darà lo stesso *rate* in presenza di problemi di arrotondamento.

«Batched bandit problems» di Perchet, Rigollet, Chassang e Snowberg[4], come segue: sia

$$u_1 = a, \quad u_m = a\sqrt{u_{m-1}}, \quad m = 2, \cdots, M, \quad t_m = \lfloor u_m \rfloor, \quad m \in [M]$$
$$u'_1 = b, \quad u'_m = bu'_{m-1}, \quad m = 2, \cdots, M, \quad t'_m = \lfloor u'_m \rfloor, \quad m \in [M]$$

dove i parametri a, b sono scelti appropriatamente tale che $t_M = t'_M = T$, i.e.,

$$a = \Theta\left(T^{\frac{1}{2-2^{1-M}}}\right), \quad b = \Theta\left(T^{\frac{1}{M}}\right) \qquad \text{Espressione 1}$$

```
Input:
    Arms 𝓘 = [K];
    orizzonte temporale T;
    numero di batches M;
    grid T = {t_1, ..., t_M};
    parametro di tuning γ.

Inizializzazione:
    𝓐 ← 𝓘.

Algoritmo:
    for m ← 1 to M − 1 do
        (a) Durante il periodo [t_{m−1} + 1, t_m], tirare un
            arm da 𝓐 lo stesso numero di volte.
        (b) Al tempo t_m:
        Sia Ȳ^max(t_m) = max_{j∈𝓐} Ȳ^j(t_m), e τ_m sia il numero
            totale di tirate di ciascuno degli arm in 𝓐.
        for i ∈ 𝓐 do
            if Ȳ^max(t_m) − Ȳ^i(t_m) ≥ √(γ log TK/τ_m) then
                𝓐 ← 𝓐 − {i}.
            end
        end
    end
    for t ← t_{M−1} + 1 to T do
        tirare l'arm i_0 tale che i_0 ∈ arg max_{j∈𝓐} Ȳ^j(t_{M−1})
            (rompere arbitrariamente i legami).
    end

Output:
    policy risultante π.
```

Frammento di codice 4.1: Pseudocodice dell'algoritmo di *Batched Successive Elimination*

Per ridurre al minimo il *minimax regret*, utilizziamo la "*minimax*" *grid* definita da $\mathcal{T}_{\text{minimax}} = \{t_1, \cdots, t_M\}$; per quanto riguarda il *regret dipendente dal problema*, usiamo la *grid* "geometrica"

che è definita da $\mathcal{T}_{\text{geometric}} = \{t'_1, \cdots, t'_M\}$. Si indicherà con π^1_{BaSE} e π^2_{BaSE} le rispettive *policies* sotto queste *grids*.

4.3.2 Analisi del *regret*

Le *performance* della BaSE *policy* sono riassunte nel seguente teorema.

Teorema 4 (*M-batched K-armed Bandit* problem, *BaSE policy*) *Si considera un problema di M-batched K-armed Bandit l'orizzonte temporale è T. Sia* π^1_{BaSE} *la BaSE policy fornita con la grid* $\mathcal{T}_{\text{minimax}}$ *e sia* π^2_{BaSE} *la BaSE policy fornita con la grid* $\mathcal{T}_{geometric}$. *Per* $\gamma \geq 12$ *e* $\max_{i\in[K]} \Delta_i = O(\sqrt{K})$, *si ha*

$$\mathbb{E}\left[R_T\left(\pi^1_{\text{BaSE}}\right)\right] \leq C \log K \sqrt{\log(KT)} \cdot \sqrt{K} T^{\frac{1}{2-2^{1-M}}} \qquad \text{Espressione 2}$$

$$\mathbb{E}\left[R_T\left(\pi^2_{\text{BaSE}}\right)\right] \leq C \log K \log(KT) \cdot \frac{K T^{1/M}}{\min_{i\neq\star} \Delta_i} \qquad \text{Espressione 3}$$

dove $C > 0$ *è una costante numerica independente da* K, M *e* T

Si noti che Teorema 4 implica il Teorema 1. Nel *sequel* si abbozza la dimostrazione del Teorema 4, in cui la principale difficoltà tecnica è controllare in modo appropriato il numero di tirate per ciascun *arm* sotto vincoli di *batch*, dove a partire dal secondo *batch* c'è un numero casuale di *arms* attivi in $\mathcal{A}$. Ci si riferisce anche a un recente lavoro («Regret Bounds for Batched Bandits» di Esfandiari, Karbasi, Mehrabian e Mirrokni[37]) per un *bound* più stretto al regret dipendente dal problema con una *grid adattabile*.

Dimostrazione 1 (del Teorema 4) *Per semplicità di notazione si suppone che ci siano* $K+1$ *arms, dove l'arm* 0 *è l'arm con il reward atteso più alto (indicato come* $\star$*), e* $\Delta_i = \mu_x - \mu_i \geq 0$ *per* $i \in [K]$. *Definire i seguenti eventi: per* $i \in [K]$, *sia* A_i *l'evento che l'arm i si elimini prima del tempo* t_{m_i}, *dove*

$$m_i = \min\left\{ j \in [M] : \text{arm } i \text{ è stato tirato almeno } \tau_i^\star \triangleq \frac{4\gamma \log(TK)}{\Delta_i^2} \text{ volte prima di } t_j \in \mathcal{T} \right\}$$

A patto che il minimo non esista, si impone $m_i = M$ e si verifica l'evento A_i. Sia B l'evento in cui l'*arm* $\star$ non viene eliminato durante l'orizzonte temporale T. L'ultimo "buon evento" E è definito come $E = \left(\cap_{i=1}^K A_i\right) \cap B$. Si nota che m_i è una variabile casuale a seconda dell'ordine in cui vengono eliminati gli *arms*. Il lemma seguente mostra che con la scelta di $\gamma \geq 12$, l'evento buono E si verifica con alta probabilità.

Lemma 5 (Probabilità di verifica dell'evento E) *L'evento E si verifica con probabilità almeno* $1 - \frac{2}{TK}$

La dimostrazione del Lemma 5 è mostrata nei materiali supplementari del *paper* oggetto di studio. Secondo il Lemma 5, il *regret* atteso $R_T(\pi)$ (con $\pi = \pi^1_{\text{BaSE}}$ oppure π^2_{BaSE}) quando l'evento E non si verifica è al massimo

$$\mathbb{E}\left[R_T(\pi)\mathbb{1}\left(E^c\right)\right] \leq T \max_{i\in[K]} \Delta_i \cdot \mathbb{P}\left(E^c\right) = O(1). \qquad \text{Espressione 4}$$

Successivamente si condiziona l'evento E e l'*upper bound* del *regret* $\mathbb{E}\left[R_T\left(\pi^1_{\text{BaSE}}\right)\ \mathbb{1}(E)\right]$ per la *minimax grid* $\mathcal{T}_{\text{minimax}}$. L'analisi della *grid* geometrica $\mathcal{T}_{\text{geometric}}$ è del tutto analoga ed è mostrata nei materiali supplementari dello studio.

Per la *policy* π^1_{BaSE}, sia $\mathcal{I}_0 \subseteq \mathcal{I}$ l'insieme (casuale) degli *arms* che vengono eliminati alla fine del primo *batch*, $\mathcal{I}_1 \subseteq \mathcal{I}$ sia l'insieme (casuale) degli *arms* rimanenti che vengono eliminati prima dell'ultimo *batch* e $\mathcal{I}_2 = \mathcal{I} - \mathcal{I}_0 - \mathcal{I}_1$ sia l'insieme (casuale) degli *arms* che rimangono nell'ultimo *batch*. È chiaro che il *regret* totale subito dagli *arms* in $\mathcal{I}_0$ è al massimo $t_1 \cdot \max_{i\in[K]} \Delta_i = O(\sqrt{K}a)$, e resta da trattare gli insiemi $\mathcal{I}_1$ e $\mathcal{I}_2$ separatamente.

Per l'*arm* $i \in \mathcal{I}_1$, sia σ_i il numero (casuale) di *arms* che vengono eliminati prima dell'*arm* i. Osservare che la frazione di tirate dell'*arm* i è al massimo $\frac{1}{K-\sigma_i}$ prima che l'*arm* i venga eliminato. Inoltre, per la definizione di t_{m_i}, si deve avere

$$\tau_i^\star > \left(\text{ numero di tirate dell'}\textit{arm } i \text{ prima di } t_{m_i-1}\right) \geq \frac{t_{m_i-1}}{K} \Longrightarrow \Delta_i\sqrt{t_{m_i-1}} \leq \sqrt{4\gamma K\log(TK)}.$$

Quindi, il *regret* totale incorso tirando un *arm* $i \in \mathcal{I}_1$ è al massimo (notare che $t_j \leq 2a\sqrt{t_{j-1}}$ per ogni $j = 2, 3, \cdots, M$ per la scelta della *grid*)

$$\Delta_i \cdot \frac{t_{m_i}}{K-\sigma_i} \leq \Delta_i \cdot \frac{2a\sqrt{t_{m_i-1}}}{K-\sigma_i} \leq \frac{2a\sqrt{4\gamma K\log(TK)}}{K-\sigma_i}$$

Notare che ci sono al massimo t elementi in $(\sigma_i : i \in \mathcal{I}_1)$ che sono almeno $K - t$ per ogni $t = 2, \cdots, K$, il *regret* totale incorso tirando gli *arms* in $\mathcal{I}_1$ è al massimo

$$\sum_{i\in\mathcal{I}_1} \frac{2a\sqrt{4\gamma K\log(TK)}}{K-\sigma_i} \leq 2a\sqrt{4\gamma K\log(TK)} \cdot \sum_{t=2}^{K} \frac{1}{t} \leq 2a\log K\sqrt{4\gamma K\log(TK)}. \qquad \text{Espressione 5}$$

Per ogni *arm* $i \in \mathcal{I}_2$, dall'analisi precedente si sa che $\Delta_i\sqrt{t_{M-1}} \leq \sqrt{4\gamma K\log(TK)}$. Quindi, sia T_i il numero di tiri dell'*arm* i, il *regret* totale incorso tirando l'*arm* $i \in \mathcal{I}_2$ è al massimo

$$\Delta_i T_i \leq T_i\sqrt{\frac{4\gamma K\log(TK)}{t_{M-1}}} \leq \frac{T_i}{T} \cdot 2a\sqrt{4\gamma K\log(TK)}.$$

dove nell'ultimo passaggio si è usato usato $T = t_M \leq 2a\sqrt{t_{M-1}}$ nella *minimax grid* $\mathcal{T}_{\text{minimax}}$. Poiché $\sum_{i \in \mathcal{I}_2} T_i \leq T$, il *regret* totale incorso tirando gli *arms* in $\mathcal{I}_2$ è al massimo

$$\sum_{i \in \mathcal{I}_2} \frac{T_i}{T} \cdot 2a\sqrt{4\gamma K \log(TK)} \leq 2a\sqrt{4\gamma K \log(TK)} \qquad \text{Espressione 6}$$

Per l'Espressione 5 e l'Espressione 6 la disuguaglianza

$$R_T\left(\pi^1_{\text{BaSE}}\right) \mathbb{1}(E) \leq 2a\sqrt{4\gamma K \log(TK)}(\log K + 1) + O(\sqrt{K}a)$$

è valida quasi sicuramente. Quindi, questa disuguaglianza combinata con l'Espressione 4 e la scelta di a nell' Espressione 1 produce l'*upper bound* (Espressione 2).

4.4 Il *Lower bound*

In questa sezione si presentano i *lower bounds* per il problema del *Batched Multi-armed Bandit*, mentra nella sottosezione 4.4.1 si disegna un problema di verifica di multi-ipotesi fisse per mostrare il *lower bound* per qualsiasi *policy* in *grid statiche*, mentre nella sottosezione 4.4.2 si costruisono diverse ipotesi per diverse *policies* sotto *grid adattabili* generali.

4.4.1 La *grid statica*

La dimostrazione del Teorema 2 fa affidamento al seguente lemma:

Lemma 6 (Il *minimax lower bound* per le *grid statiche*) *Per qualsiasi grid statica* $0 = t_0 < t_1 < \cdots < t_M = T$ *e il minimo gap* $\Delta \in (0, \sqrt{K})$*, il seguente minimax lower bound vale per qualsiasi policy* π *sotto questa grid:*

$$\sup_{\{\mu^{(i)}\}_{i=1}^K : \Delta_i \in \{0\} \cup [\Delta, \sqrt{K}]} \mathbb{E}\left[R_T(\pi)\right] \geq \Delta \cdot \sum_{j=1}^{M} \frac{t_j - t_{j-1}}{4} \exp\left(-\frac{2t_{j-1}\Delta^2}{K-1}\right).$$

Per prima cosa si mostra che Lemma 6 implica Teorema 2 scegliendo il *gap* più piccolo $\Delta > 0$ in modo appropriato. Secondo le definizioni del *minimax regret* $R^\star_{\text{min}-\text{max}}$ e del *regret dipendente dal problema* $R^\star_{\text{pro-dep}}$, scegliere $\Delta = \Delta_j = \sqrt{(K-1)/\left(t_{j-1}+1\right)} \in [0, \sqrt{K}]$ in Lemma 6 produce che

$$R^\star_{\text{min}-\text{max}}(K, M, T) \geq c_0\sqrt{K} \cdot \max_{j \in [M]} \frac{t_j}{\sqrt{t_{j-1}+1}},$$

$$R^\star_{\text{pro-dep}}(K, M, T) \geq c_0 K \cdot \max_{j \in [M]} \frac{t_j}{t_{j-1}+1},$$

per qualche costante numerica $c_0 > 0$. Dato che $t_0 = 0$, $t_M = T$, i *lower bounds* nel Teorema 2 reggono.

Successivamente si utilizza l'idea generale del testing di ipotesi multiple per dimostrare il Lemma 6. Considerare le seguenti K distribuzioni *candidate* di *reward*:

$$\begin{aligned}
P_1 &= \mathcal{N}(\Delta, 1) \otimes \mathcal{N}(0, 1) \otimes \mathcal{N}(0, 1) \otimes \cdots \otimes \mathcal{N}(0, 1), \\
P_2 &= \mathcal{N}(\Delta, 1) \otimes \mathcal{N}(2\Delta, 1) \otimes \mathcal{N}(0, 1) \otimes \cdots \otimes \mathcal{N}(0, 1), \\
P_3 &= \mathcal{N}(\Delta, 1) \otimes \mathcal{N}(0, 1) \otimes \mathcal{N}(2\Delta, 1) \otimes \cdots \otimes \mathcal{N}(0, 1), \\
&\vdots \\
P_K &= \mathcal{N}(\Delta, 1) \otimes \mathcal{N}(0, 1) \otimes \mathcal{N}(0, 1) \otimes \cdots \otimes \mathcal{N}(2\Delta, 1) -
\end{aligned}$$

Si osserva che questa costruzione non è del tutto simmetrica, dove la distribuzione del *reward* del primo *arm* è sempre $\mathcal{N}(\Delta, 1)$. Le proprietà chiave di questa costruzione sono le seguenti:

1. Per ogni $i \in [K]$, l'*arm* i è l'*arm* ottimale sotto la distribuzione del *reward* P_i
2. Per ogni $i \in [K]$, tirare un *arm* sbagliato comporta un *regret* di almeno Δ sotto la distribuzione del *reward* P_i

Di conseguenza, poiché il *regret* medio serve come *lower bound* del *regret* nel caso peggiore, si ha:

$$\sup_{\{\mu^{(i)}\}_{i=1}^{K} : \Delta_i \in \{0\} \cup [\Delta, \sqrt{K}]} \mathbb{E} R_T(\pi) \geq \frac{1}{K} \sum_{i=1}^{K} \sum_{t=1}^{T} \mathbb{E}_{P_i^t} R^t(\pi) \geq \Delta \sum_{t=1}^{T} \frac{1}{K} \sum_{i=1}^{K} P_i^t(\pi_t \neq i) \quad \text{Espressione 7}$$

dove P_i^t indica la distribuzione delle osservazioni disponibili al momento t sotto P_i, e $R^t(\pi)$ indica il *regret* istantaneo incorso dalla *policy* π_t al momento t. Quindi, rimane da limitare inferiormente la quantità $\frac{1}{K} \sum_{i=1}^{K} P_i^t(\pi_t \neq i)$ per qualsiasi $t \in [T]$, che è l'oggetto del seguente lemma:

Lemma 7 (Misure di probabilità su alcuni spazi comuni di probabilità) *Siano $Q_1, \cdots, Q_n$ le misure di probabilità su alcuni spazi comuni di probabilità $(\Omega, \mathcal{F})$, e $\Psi : \Omega \to [n]$ sia qualsiasi funzione misurabile. Per qualsiasi albero $T = ([n], E)$ con insieme di* vertexes *$[n]$ e insieme di* edges *E si ha*

$$\frac{1}{n} \sum_{i=1}^{n} Q_i(\Psi \neq i) \geq \sum_{(i,j) \in E} \frac{1}{2n} \exp\left(-D_{\mathrm{KL}}\left(Q_i \| Q_j\right)\right).$$

La dimostrazione del Lemma 7 è rinviata ai materiali supplementari dello studio e di seguito si riportano solo alcune osservazioni.

Osservazione 1 (Limite inferiore, la disuguaglianza di Fano e dipendenza da KL) *Un limite inferiore più noto per $\frac{1}{n} \sum_{i=1}^{n} Q_i(\Psi \neq i)$ è la disuguaglianza di Fano[38], che coinvolge le informazioni reciproche $I(U; X)$ con $U \sim \mathrm{Uniform}([n])$ e $P_{X|U=i} = Q_i$. Tuttavia siccome $I(U; X) = \mathbb{E}_{P_U} D_{\mathrm{KL}}\left(P_{X|U} \| P_X\right)$, la disuguaglianza di Fano dà un limite inferiore che dipende linearmente sulla divergenza di coppia KL anziché esponenzialmente e quindi è rilassata per lo scopo che si intende.*

Osservazione 2 (Limite inferiore alternativo con somma viene presa su tutte le coppie) *Un limite inferiore alternativo è utilizzare* $\frac{1}{2n^2}\sum_{i\neq j}\exp\left(-D_{\mathrm{KL}}\left(Q_i\|Q_j\right)\right)$*, i.e. la somma viene presa su tutte le coppie* (i,j) *anziché solo gli edges di un albero. Tuttavia, questo limite è più debole di Lemma 7 e nel caso in cui* $Q_i = \mathcal{N}(i\Delta,1)$ *per un grande* $\Delta > 0$*, Lemma 7 con l'albero* $T = ([n],\{(1,2),(2,3),\cdots,(n-1,n)\})$ *è stretto (dando il rate* $\left(\exp\left(-O\left(\Delta^2\right)\right)\right)$ *mentre il limite alternativo perde un fattore di n (dando il rate* $\exp\left(-O\left(\Delta^2\right)\right)/n$*).*

Al limite inferiore Espressione 7, si applica Lemma 7 con l'albero stella $T = ([n],\{(1,i) : 2 \leq i \leq n\})$. Per $i \in [K]$, si indica per $T_i(t)$ il numero di tiri dell'*arm i* prima all'attuale *batch* di t. Quindi, $\sum_{i=1}^{K} T_i(t) = t_{j-1}$ se $t \in \left(t_{j-1}, t_j\right]$. Inoltre, poiché $D_{\mathrm{KL}}\left(P_1^t\|P_i^t\right) = 2\Delta^2\mathbb{E}_{P_1^t}T_i(t)$, si ha

$$\begin{aligned}\frac{1}{K}\sum_{i=1}^{K}P_i^t\left(\pi_t\neq i\right) &\geq \frac{1}{2K}\sum_{i=2}^{K}\exp\left(-D_{\mathrm{KL}}\left(P_1^t\|P_i^t\right)\right) = \frac{1}{2K}\sum_{i=2}^{K}\exp\left(-2\Delta^2\mathbb{E}_{P_1^t}T_i(t)\right)\\ &\geq \frac{K-1}{2K}\exp\left(-\frac{2\Delta^2}{K-1}\mathbb{E}_{P_1^t}\sum_{i=2}^{K}T_i(t)\right) \geq \frac{1}{4}\exp\left(-\frac{2\Delta^2 t_{j-1}}{K-1}\right)\end{aligned} \qquad \text{Espressione 8}$$

Combinando l'Espressione 7 e l'Espressione 8 si completa la dimostrazione del Lemma 6.

4.4.2 La *grid adattabile*

Si esamina il caso in cui la *grid* può essere randomizzata e generata sequenzialmente in modo adattivo. Si ricorda che nella sezione precedente, si sono costruite più ipotesi fisse e si è dimostrato che nessuna *policy* in una *grid statica* può ottenere un *regret* uniformemente piccolo in tutte le ipotesi. Tuttavia, questo argomento cade anche se la *grid* è solo randomizzata ma non adattabile, a causa della natura non convessa del limite inferiore in Lemma 6 (in $(t_1,\cdots,t_M)$). In altre parole, non si può sperare che un singolo problema fisso di verifica delle ipotesi multiple funzioni per tutte le *policies*. Per superare questa difficoltà, una subroutine nella dimostrazione del Teorema 3 è quella di costruire ipotesi appropriate dopo che la *policy* è stata data (confrontare con la dimostrazione di Lemma 8. La dimostrazione viene fatta di seguito.

Si dimostra solo il limite inferiore per il *minimax regret*, mentre l'analisi del *regret dipendente dal problema* è del tutto analoga. Si consideri il seguente tempo $T_1,\cdots,T_M \in [1,T]$ e i *gaps* $\Delta_1,\cdots,\Delta_M \in (0,\sqrt{K}]$ con

$$T_j = \left\lfloor T^{\frac{1-2^{-j}}{1-2^{-1}}}\right\rfloor, \quad \Delta_j = \frac{\sqrt{K}}{36M}\cdot T^{-\frac{1-2^{1-j}}{2\left(1-2^{-11}\right)}}, \quad j\in[M] \qquad \text{Espressione 9}$$

Sia $\mathcal{T} = \{t_1,\cdots,t_M\}$ una qualsiasi *grid adattabile* e π sia qualsiasi *policy* sotto la *grid* $\mathcal{T}$. Per ogni $j \in [M]$, si definisce l'evento $A_j = \left\{t_{j-1} < T_{j-1}, t_j \geq T_j\right\}$ sotto la *policy* π con la convenzione

che $t_0 = 0, t_M = T$. Si noti che gli eventi $A_1, \cdots, A_M$ formano una partizione dell'intero spazio di probabilità. Si definisce anche la seguente famiglia di distribuzioni di *rewards*: per $j \in [M-1]$, $k \in [K-1]$ sia

$$P_{j,k} = \mathcal{N}(0,1) \otimes \cdots \otimes \mathcal{N}(0,1) \otimes \mathcal{N}\left(\Delta_j + \Delta_M, 1\right) \otimes \mathcal{N}(0,1) \otimes \cdots \otimes \mathcal{N}(0,1) \otimes \mathcal{N}\left(\Delta_M, 1\right)$$

dove k-esimo componente di $P_{j,k}$ ha una media diversa da zero. Per $j = M$, si definisce

$$P_M = \mathcal{N}(0,1) \otimes \cdots \otimes \mathcal{N}(0,1) \otimes \mathcal{N}\left(\Delta_M, 1\right)$$

Si noti che questa costruzione garantisce che $P_{j,k}$ e P_M differiscano solo nella componente k-esimo, che è cruciale per i risultati di indistinguibilità in Lemma 9.

Si è interessati alle seguenti quantità:

$$p_j = \frac{1}{K-1} \sum_{k=1}^{K-1} P_{j,k}\left(A_j\right), \quad j \in [M-1], \quad p_M = P_M\left(A_M\right)$$

dove $P_{j,k}(A)$ indica la probabilità dell'evento A data la distribuzione vera del *reward* $P_{j,k}$ e la *policy* π. L'importanza di queste quantità sta nei seguenti lemmi.

Lemma 8 (Se un evento A_j succede con probabilità non piccola, π ha un *regret* grande nel caso peggiore) *Se $p_j \geq \frac{1}{2M}$ per una $j \in [M]$, allora si ha*

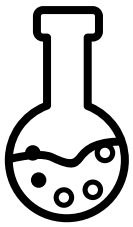
$$\sup_{\{\mu^{(i)}\}_{i=1}^{K}: \Delta_i \leq \sqrt{K}} \mathbb{E}\left[R_T(\pi)\right] \geq cM^{-2} \cdot \sqrt{K} T^{\frac{1}{2-2}!-M},$$

dove $c > 0$ è una costante numerica indipendente da (K, M, T) e $(\pi, \mathcal{T})$.

Lemma 9 (Almeno un p_j dovrebbe essere grande) *Vale la disuguaglianza: $\sum_{j=1}^{M} p_j \geq \frac{1}{2}$*

Le dimostrazioni dettagliate di Lemma 8 e Lemma 9 sono rinviate ai materiali supplementari dello studio analizzato, e qui si abbozzano solo le idee. Lemma 8 afferma che, se uno qualsiasi degli eventi A_j si verifica con una probabilità non piccola nel rispettivo j-esimo mondo (i.e., sotto la combinazione di $\left(P_{j,k} : k \in [K-1]\right)$ o P_M), allora la *policy* π ha un grande *regret* nel caso peggiore. L'intuizione dietro il Lemma 8 è che, se l'evento $t_{j-1} \leq T_{j-1}$ si verifica sotto la distribuzione della *reward* $P_{j,k}$, allora le osservazioni nei primi $(j-1)$ *batches* non sono sufficienti per distinguere $P_{j,k}$ dalla sua versione perturbata (attentamente progettata) con dimensioni di perturbazione Δ_j. Inoltre, se in aggiunta vale $t_j \geq T_j$, allora il *regret* totale è almeno $\Omega\left(T_j \Delta_j\right)$ a causa dell'indistinguibilità delle Δ_j perturbazioni nei primi j *batches*. Quindi, se A_j si presenta con una probabilità abbastanza grande, anche il *regret* totale risultante sarà grande.

Il Lemma 9 è complementare al (o completa il) Lemma 8 affermando che almeno un p_j dovrebbe essere grande. Si noti che se tutti p_j fossero definiti nello stesso mondo, la struttura della partizione di $A_1, \cdots, A_M$ implicherebbe $\sum_{j\in[M]} p_j \geq 1$. Poiché il verificarsi di A_j non può davvero aiutare a distinguere il j-esimo mondo da quelli successivi, Lemma 9 mostra che si potrà ancora operare nello stesso mondo e arrivare a una costante leggermente più piccola di 1. Infine si mostra come Lemma 8 e Lemma 9 implicano il Teorema 2. In effetti, da Lemma 9 esistono alcuni $j \in [M]$ tali che $p_j \geq (2M)^{-1}$. Quindi con Lemma 8 e l'arbitrarietà di π si arriva al limite inferiore desiderato nel Teorema 2.



Capitolo 5

Esperimenti

Icona del Capitolo 5[2]

5.1 Esperimenti già eseguiti dagli autori

Gli autori dello studio in oggetto hanno eseguito alcuni esperimenti usando la BaSE *policy* proposta sotto diverse *grids* facendo confronti con le *performance* di altri algoritmi. Gli esperimenti di valutazione empirica delle *performance* dei *regrets* sono di 4 tipi:

- *regret* medio vs. il numero dei *batches* M (per *minimax grid*, *geometric grid*, *arithmetic grid* e *UCB1*)
- *regret* medio vs. il numero degli *arms* K (per *minimax grid*, *geometric grid*, *arithmetic grid* e *UCB1*)
- *regret* medio vs. l'orizzonte temporale T (per *minimax grid*, *geometric grid*, *arithmetic grid* e *UCB1*)
- *regret* medio vs. il numero dei *batches* M come confronto tra BaSE (per *minimax grid* e *geometric grid*) vs. ETC (per *minimax grid* e *geometric grid*) vs. *UCB1*)

5.1.1 Dati e parametri

I dati sono generati a *random* fornendo valori come limite inferiore e limite superiore e le dimensioni dei vettori o delle matrici da riempire con questi dati.

I parametri predefinito sono $T = 5 \times 10^4$, $K = 3$, $M = 3$ e $\gamma = 1$, e il *reward* medio è $\mu^\star = 0.6$ per l'*arm* ottimale ed è $\mu = 0,5$ per tutti gli altri *arms*. Oltre alle *minimax grids* e alle *geometric grids*, si sperimenta anche sulla *arithmetic grid* con $t_j = jT/M$ per $j \in [M]$.

5.1.2 Codice originale Matlab

I codici sorgente dell'esperimento sono forniti apertamente dagli autori in https://github.com/Mathegineer/batched-bandit/[39]. Il codice è scritto in Matlab. Segue il codice:

Codice sorgente originale in Matlab, *file*: `main.m`:

```
clear;clc;close all;

%% Parameters
K = 3; T = 5e4; M = 3;
K_set = round(logspace(log10(2),log10(20),6));
T_set = round(logspace(log10(500),log10(5e4),6));
M_set = 2:7;
mu_max = 0.6; mu_min = 0.5; gamma = 0.5; m = 200;

%% Experiments

% dependence on M
regretMinimax_M = zeros(m,length(M_set));
regretGeometric_M = zeros(m,length(M_set));
regretArithmetic_M = zeros(m,length(M_set));
regretUCB_M = zeros(m,1);
mu = [mu_max, mu_min * ones(1,K-1)];
for iter = 1 : m
    regretUCB_M(iter) = UCB1(mu,K,T);
    for iter_M = 1 : length(M_set)
        temp_M = M_set(iter_M);
        regretMinimax_M(iter,iter_M) = BASEFunc(mu,K,T,temp_M,'minimax',gamma);
        regretGeometric_M(iter,iter_M) = BASEFunc(mu,K,T,temp_M,'geometric',
    gamma);
        regretArithmetic_M(iter,iter_M) = BASEFunc(mu,K,T,temp_M,'arithmetic',
    gamma);
    end
end
regretMinimax_M_mean = mean(regretMinimax_M) / T;
regretGeometric_M_mean = mean(regretGeometric_M) / T;
regretArithmetic_M_mean = mean(regretArithmetic_M) / T;
regretUCB_M_mean = mean(regretUCB_M) / T;

% dependence on K
regretMinimax_K = zeros(m,length(K_set));
```

[39] Zijun Gao, Yanjun Han, Zhimei Ren e Zhengqing Zhou. *Batched Multi-armed Bandits Problem Source code*. Inglese. Source code of the experiments. 2020. URL: https://github.com/Mathegineer/batched-bandit/.

```
regretGeometric_K = zeros(m,length(K_set));
regretArithmetic_K = zeros(m,length(K_set));
regretUCB_K = zeros(m,length(K_set));
for iter = 1 : m
    for iter_K = 1 : length(K_set)
        temp_K = K_set(iter_K);
        mu = [mu_max, mu_min * ones(1, temp_K-1)];
        regretUCB_K(iter,iter_K) = UCB1(mu,temp_K,T);
        regretMinimax_K(iter,iter_K) = BASEFunc(mu,temp_K,T,M,'minimax',gamma);
        regretGeometric_K(iter,iter_K) = BASEFunc(mu,temp_K,T,M,'geometric',
    gamma);
        regretArithmetic_K(iter,iter_K) = BASEFunc(mu,temp_K,T,M,'arithmetic',
    gamma);
    end
end
regretMinimax_K_mean = mean(regretMinimax_K) / T;
regretGeometric_K_mean = mean(regretGeometric_K) / T;
regretArithmetic_K_mean = mean(regretArithmetic_K) / T;
regretUCB_K_mean = mean(regretUCB_K) / T;

% dependence on T
regretMinimax_T = zeros(m,length(T_set));
regretGeometric_T = zeros(m,length(T_set));
regretArithmetic_T = zeros(m,length(T_set));
regretUCB_T = zeros(m,length(T_set));
mu = [mu_max, mu_min * ones(1, K-1)];
for iter = 1 : m
    for iter_T = 1 : length(T_set)
        temp_T = T_set(iter_T);
        regretUCB_T(iter,iter_T) = UCB1(mu,K,temp_T)/temp_T;
        regretMinimax_T(iter,iter_T) = BASEFunc(mu,K,temp_T,M,'minimax',gamma)/
    temp_T;
        regretGeometric_T(iter,iter_T) = BASEFunc(mu,K,temp_T,M,'geometric',
    gamma)/temp_T;
        regretArithmetic_T(iter,iter_T) = BASEFunc(mu,K,temp_T,M,'arithmetic',
    gamma)/temp_T;
    end
end
regretMinimax_T_mean = mean(regretMinimax_T);
regretGeometric_T_mean = mean(regretGeometric_T);
regretArithmetic_T_mean = mean(regretArithmetic_T);
regretUCB_T_mean = mean(regretUCB_T);
```

```
% comparison with [PRCS16]
regretMinimax = zeros(m,length(M_set));
regretGeometric = zeros(m,length(M_set));
regretPRCSminimax = zeros(m,length(M_set));
regretPRCSgeometric = zeros(m,length(M_set));
regretUCB = zeros(m,1);
mu = [mu_max, mu_min];
for iter = 1 : m
    regretUCB(iter) = UCB1(mu,2,T);
    for iter_M = 1 : length(M_set)
        temp_M = M_set(iter_M);
        regretMinimax(iter,iter_M) = BASEFunc(mu,2,T,temp_M,'minimax',gamma);
        regretGeometric(iter,iter_M) = BASEFunc(mu,2,T,temp_M,'geometric',gamma
    );

        regretPRCSminimax(iter,iter_M) = PRCS_twoarm(mu,temp_M,T,'minimax');
        regretPRCSgeometric(iter,iter_M) = PRCS_twoarm(mu,temp_M,T,'geometric')
    ;
    end
end
regretMinimax_mean = mean(regretMinimax) / T;
regretGeometric_mean = mean(regretGeometric) / T;
regretPRCSminimax_mean = mean(regretPRCSminimax) / T;
regretPRCSGeometric_mean = mean(regretPRCSgeometric) / T;
regretUCB_mean = mean(regretUCB) / T;

% Figures
figure;
plot(M_set, regretMinimax_M_mean, 'bs-', 'MarkerFaceColor','b','linewidth', 2);
     hold on;
plot(M_set, regretGeometric_M_mean, 'ro--', 'MarkerFaceColor','r','linewidth',
    2);
plot(M_set, regretArithmetic_M_mean, 'cv-.', 'MarkerFaceColor','c','linewidth'
    ,2);
plot(M_set, regretUCB_M_mean * ones(size(M_set)), 'k:', 'linewidth', 2);
xticks(2:7); xlabel('M'); ylabel('Average regret');
legend('Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1');

figure;
plot(K_set, regretMinimax_K_mean, 'bs-', 'MarkerFaceColor','b','linewidth', 2);
     hold on;
plot(K_set, regretGeometric_K_mean, 'ro--', 'MarkerFaceColor','r','linewidth',
    2);
```

```
plot(K_set, regretArithmetic_K_mean, 'cv-.', 'MarkerFaceColor','c','linewidth'
    ,2);
plot(K_set, regretUCB_K_mean, 'k:', 'linewidth', 2);
xlabel('K'); ylabel('Average regret');
legend('Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1');

figure;
semilogx(T_set, regretMinimax_T_mean, 'bs-', 'MarkerFaceColor','b','linewidth',
     2); hold on;
semilogx(T_set, regretGeometric_T_mean, 'ro--', 'MarkerFaceColor','r','
    linewidth', 2);
semilogx(T_set, regretArithmetic_T_mean, 'cv-.', 'MarkerFaceColor','c','
    linewidth',2);
semilogx(T_set, regretUCB_T_mean, 'k:', 'linewidth', 2);
xlim([5e2,5e4]); xlabel('T'); ylabel('Average regret');
legend('Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1');

figure;
plot(M_set, regretMinimax_mean, 'bs-', 'MarkerFaceColor','b','linewidth', 2);
    hold on;
plot(M_set, regretGeometric_mean, 'bs--', 'MarkerFaceColor','b','linewidth', 2)
    ;
plot(M_set, regretPRCSminimax_mean, 'ro-', 'MarkerFaceColor','r','linewidth',
    2);
plot(M_set, regretPRCSGeometric_mean, 'ro--', 'MarkerFaceColor','r','linewidth'
    , 2);
plot(M_set, regretUCB_mean * ones(size(M_set)), 'k:', 'linewidth', 2);
xticks(2:7); xlabel('M'); ylabel('Average regret');
legend('BaSE: Minimax Grid', 'BaSE: Geometric Grid', 'ETC: Minimax Grid', 'ETC:
     Geometric Grid', 'UCB1');
```

Frammento di codice 5.1: Codice sorgente originale in Matlab, *file*: `main.m`

Codice sorgente originale in Matlab, *file*: `BASEFunc.m`:

```
% function of BASE
% parameters
%     K: number of batches
%     TSeq: horizon
%     M: number of batches
%     b = T^(1/M); TGridAdaptive = floor(b.^(1:M));...,
%         TGridAdaptive = floor(TGridAdaptive/K) * K; TGridAdaptive(M) = T;
    ...,
%         TGridAdaptive = [0,TGridAdaptive]; % adaptive batch grids
%     a = T^(1/(2 - 2^(1-M))); TGridMinimax = floor(a.^(2.-1./2.^(0:M-1)));...,
```

```
%     TGridMinimax(M) = T; ...,
%     TGridMinimax = [0,TGridMinimax]; % minimax batch grids
%     mu: batch mean
%     gamma: tunning parameter

function [regret, activeSet] = BASEFunc(mu, K, T, M, gridType, gamma)
    % record
    regret = 0;
    if strcmp(gridType,'minimax')
        a = T^(1/(2 - 2^(1-M))); TGrid = floor(a.^(2.-1./2.^(0:M-1)));...,
        TGrid(M) = T; TGrid = [0,TGrid]; % minimax batch grids
    elseif strcmp(gridType,'geometric')
        b = T^(1/M); TGrid = floor(b.^(1:M)); TGrid(M) = T; ...,
        TGrid = [0,TGrid]; % geometric batch grids
    elseif strcmp(gridType,'arithmetic')
        TGrid = floor(linspace(0, T, M+1));
    end

    % initialization
    activeSet = ones(1,K); numberPull = zeros(1,K); averageReward = zeros(1,K);

    for i = 2:M+1
        availableK = sum(activeSet);
        pullNumber = max(floor((TGrid(i) - TGrid(i-1))/availableK), 1);
        TGrid(i) = availableK * pullNumber + TGrid(i-1);
        for j = find(activeSet == 1)
            averageReward(j) = averageReward(j) * (numberPull(j)/(numberPull(j)
     ...,
                + pullNumber)) + (mean(randn(1,pullNumber)) + mu(j)) * ...,
                (pullNumber/(numberPull(j) + pullNumber));
            regret = regret + (pullNumber * (mu(1) - mu(j)));
            numberPull(j) = numberPull(j) + pullNumber;
        end
        maxArm = max(averageReward(find(activeSet == 1)));
        for j = find(activeSet == 1)
            if ((maxArm - averageReward(j)) >= sqrt(gamma * log(T*K) /
    numberPull(j)))
                activeSet(j) = 0;
            end
        end
    end
end
```

Frammento di codice 5.2: Codice sorgente originale in Matlab, *file*: `BASEFunc.m`

Codice sorgente originale in Matlab, *file*: `UCB1.m`:

```
function regret = UCB1(mu,K,T)
    pullnumber = ones(K,1);
    averageReward = zeros(K,1);

    for t = 1 : K
        averageReward(t) = mu(t) + randn;
    end

    for t = (K+1) : T
        UCB = averageReward + sqrt(2*log(T) ./ pullnumber);
        [~, pos] = max(UCB);
        weight = 1/(pullnumber(pos) + 1);
        averageReward(pos) = (1-weight) * averageReward(pos) ...
            + weight * (mu(pos) + randn);
        pullnumber(pos) = pullnumber(pos) + 1;
    end

    regret = (mu(1) - mu(2:end)) * pullnumber(2:end);
end
```

Frammento di codice 5.3: Codice sorgente originale in Matlab, *file*: `UCB1.m`

Codice sorgente originale in Matlab, *file*: `PRCS_twoarm.m`:

```
function regret = PRCS_twoarm(mu,M,T,gridType)
    if strcmp(gridType,'minimax')
        a = T^(1/(2 - 2^(1-M))); TGrid = np.floor(a.^(2.-1./2.^(0:M-1)));...,
        TGrid(M) = T; TGrid = [0,TGrid]; % minimax batch grids
    elseif strcmp(gridType,'geometric')
        b = T^(1/M); TGrid = floor(b.^(1:M)); TGrid(M) = T; ...,
        TGrid = [0,TGrid]; % adaptive batch grids
    end

    pullnumber = round(TGrid(2)/2);
    regret = pullnumber * (mu(1) - mu(2));
    reward = sum([randn(pullnumber,1) + mu(1), randn(pullnumber,1) + mu(2) ],
    1);
    opt = 0;

    for m = 2 : M
        t = TGrid(m);
        thres = sqrt(4 * log(2*T/t) / t);
        if opt == 0
```

```
            if (reward(1) - reward(2))/t > thres
                opt = 1;
            elseif (reward(2) - reward(1))/t > thres
                opt = 2;
            else
                cur_number = round((TGrid(m+1) - TGrid(m))/2);
                pullnumber = pullnumber + cur_number;
                reward = reward + sum([randn(cur_number,1) + mu(1), ...
                    randn(cur_number,1) + mu(2) ]);
                regret = regret + cur_number * (mu(1) - mu(2));
            end
        end

        if opt == 2
            regret = regret + (TGrid(m+1) - TGrid(m)) * (mu(1) - mu(2));
        end

        if m == (M-1)
            if reward(1) > reward(2)
                opt = 1;
            else
                opt = 2;
            end
        end
    end
end
```

Frammento di codice 5.4: Codice sorgente originale in Matlab, *file*: `PRCS_twoarm.m`

5.1.3 Risultati e osservazioni

La Figura 5.1, Figura 5.2 e Figura 5.3 mostrano la dipendenza empirica dei BaSE *regrets* medi in diverse *grids*, insieme al confronto con l'algoritmo centralizzato UCB1[16] senza vincoli di *batch*.

Si osserva che nella *minimax grid* c'è in genere un minor *regret* tra tutte le *grids* e che i *batches* $M = 4$ sembrano essere sufficienti affinché le prestazioni BaSE si avvicinino alle prestazioni centralizzate. Si confronta anche l'algoritmo BaSE con l'algoritmo ETC in[4] per il caso a due *arms*, e la Figura 5.4 mostra che BaSE ottiene *regrets* più bassi di ETC.

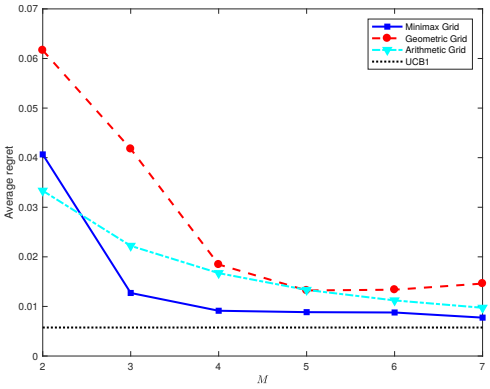


Figura 5.1: *Regret* medio vs. numero di *batches* M.

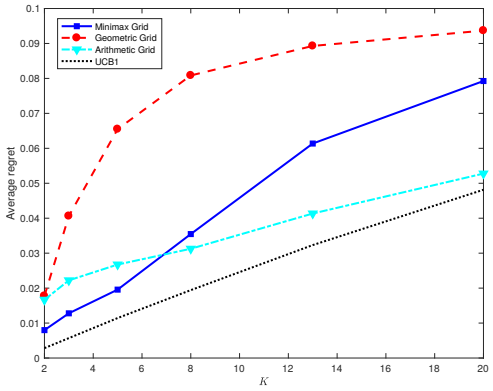


Figura 5.2: *Regret* medio vs. numero di *arms* K.

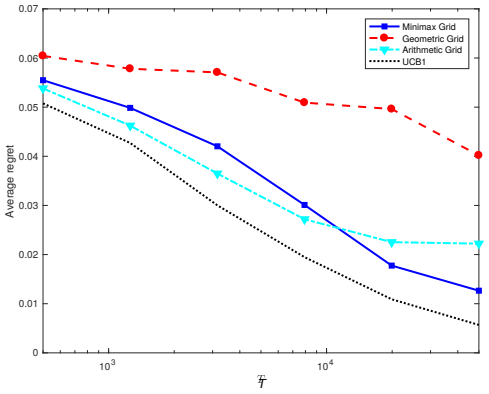


Figura 5.3: *Regret* medio vs. orizzonte temporale T.

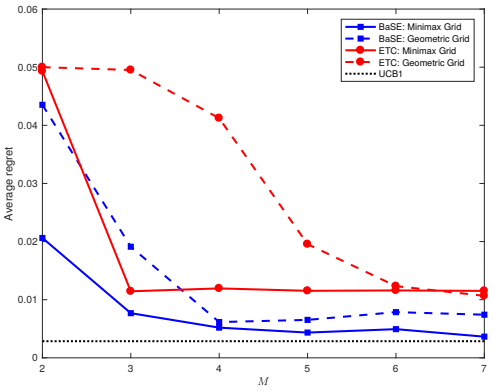


Figura 5.4: Confronto tra BaSE e ETC.

Figura 5.5: Risultati empirici delle *regret performance* degli esperimenti della BaSE *policy* pubblicati nello studio.

5.2 Nuovi esperimenti eseguiti su dati randomizzati

Gli stessi esperimenti effettuati dagli autori dello studio sono stati effettuati di nuovo, con gli stessi parametri e con dati generati casualmente. Successivamente, a partire dal codice fornito dal *paper* in Matlab, si è fatto il *transpiling* manuale del programm a Python facendo estensivo uso anche delle librarie `NumPy` e `matplotlib`. Gli stessi esperimenti di cui sopra, poi sono stati effettuati nuovamente, con gli stessi parametri e con dati generati casualmente, sia in Matlab che in Python.

5.2.1 Dati e parametri

I dati sono stati generati nello stesso modo di come sono generati i dati su cui gli autori hanno eseguito gli esperimenti, quindi generati a *random*. La modalità di generazione è la stessa, i dati ovviamente sono diversi.

I parametri sono gli stessi usati durante gli esperimenti degli autori.

5.2.2 Codice Python

I codici sorgente Matlab degli esperimenti sono gli stessi forniti dagli autori, vedere sezione precedete. Il codice in Python post-*transpiling* invece è questo:

Codice sorgente dopo *transpiling* in Python, *file*: `main.py`:

```
import math
import numpy as np

from matplotlib import pyplot as plt

from UCB1 import UCB1
from BASEFunc import BASEFunc
from PRCS_twoarm import PRCS_twoarm

"""
Parametri
"""

# K = 3
# T = 5*10^4
# M = 3
K = 3
T = int(5e4)
M = 3
```

```
# K_set = vettore riga di 6 punti da log10(2) a log10(20) ugualmente
# logaritmicamente distanti (arrotondati)
# T_set = vettore riga di 6 punti da log10(500) a log10(5*10^4) ugualmente
# logaritmicamente distanti (arrotondati)
# M_set = insieme dei punti da 2 a 7 (compreso).
K_set = np.round(np.logspace(math.log(2, 10), math.log(20, 10), num=6,
                             endpoint=True,
                             base=10.0,
                             dtype=None, axis=0))
T_set = np.round(np.logspace(math.log(500, 10), math.log(int(5e4), 10), num=6,
                             endpoint=True,
                             base=10.0,
                             dtype=None, axis=0))
M_set = range(2, 8)

# mu_max = 0.6 reward medio per l'arm ottimale
# mu_min = 0.5 reward medio per tutti gli altri arm
# gamma = 0.5
# m = 200
mu_max = 0.6
mu_min = 0.5
gamma = 0.5
m = 200

"""
Esperimenti
"""

# dipendenza da M

# regretMinimax_M = matrice di zeri di dimensione mxlen(M_set)
# regretGeometric_M = matrice di zeri di dimensione mxlen(M_set)
# regretArithmetic_M = matrice di zeri di dimensione mxlen(M_set)
# regretUCB_M = vettore di zeri di dimensione mx1
# mu = i rewards come concatenazione (bind di colonne) dei
# reward di mu_max e tutti i K-1 mu_min
regretMinimax_M = np.zeros((m, len(M_set)), dtype=float)
regretGeometric_M = np.zeros((m, len(M_set)), dtype=float)
regretArithmetic_M = np.zeros((m, len(M_set)), dtype=float)
regretUCB_M = np.zeros((m,), dtype=float)
mu = np.concatenate(([mu_max], mu_min * np.ones((1, K-1),
                                                dtype=int).flatten()), axis=0)
```

```
for iter_i in range(0, m):
    print("Calculating First  "+str(iter_i))
    regretUCB_M[iter_i] = UCB1(mu, K, T)
    for iter_M in range(0, len(M_set)):
        temp_M = M_set[iter_M]
        regretMinimax_M[iter_i, iter_M] = \
            BASEFunc(mu, K, T, temp_M, 'minimax', gamma)
        regretGeometric_M[iter_i, iter_M] = \
            BASEFunc(mu, K, T, temp_M, 'geometric', gamma)
        regretArithmetic_M[iter_i, iter_M] = \
            BASEFunc(mu, K, T, temp_M, 'arithmetic', gamma)

regretMinimax_M_mean = np.mean(regretMinimax_M, axis=0) / T
regretGeometric_M_mean = np.mean(regretGeometric_M, axis=0) / T
regretArithmetic_M_mean = np.mean(regretArithmetic_M, axis=0) / T
regretUCB_M_mean = np.mean(regretUCB_M, axis=0) / T


# dependence on K

regretMinimax_K = np.zeros((m, len(K_set)), dtype=float)
regretGeometric_K = np.zeros((m, len(K_set)), dtype=float)
regretArithmetic_K = np.zeros((m, len(K_set)), dtype=float)
regretUCB_K = np.zeros((m, len(K_set)), dtype=float)

for iter_i in range(0, m):
    print("Calculating Second "+str(iter_i))
    for iter_K in range(0, len(K_set)):
        temp_K = int(K_set[iter_K])
        mu = np.concatenate(([mu_max], mu_min *
                             np.ones((1, temp_K-1),
                                     dtype=int).flatten()), axis=0)
        regretUCB_K[iter_i, iter_K] = UCB1(mu, temp_K, T)
        regretMinimax_K[iter_i, iter_K] = \
            BASEFunc(mu, temp_K, T, M, 'minimax', gamma)
        regretGeometric_K[iter_i, iter_K] = \
            BASEFunc(mu, temp_K, T, M, 'geometric', gamma)
        regretArithmetic_K[iter_i, iter_K] = \
            BASEFunc(mu, temp_K, T, M, 'arithmetic', gamma)


regretMinimax_K_mean = np.mean(regretMinimax_K, axis=0) / T
```

```
regretGeometric_K_mean = np.mean(regretGeometric_K, axis=0) / T
regretArithmetic_K_mean = np.mean(regretArithmetic_K, axis=0) / T
regretUCB_K_mean = np.mean(regretUCB_K, axis=0) / T


# dependence on T

regretMinimax_T = np.zeros((m, len(T_set)), dtype=float)
regretGeometric_T = np.zeros((m, len(T_set)), dtype=float)
regretArithmetic_T = np.zeros((m, len(T_set)), dtype=float)
regretUCB_T = np.zeros((m, len(T_set)), dtype=float)
mu = np.concatenate(([mu_max], mu_min * np.ones((1, K-1),
                                                dtype=int).flatten()), axis=0)

for iter_i in range(0, m):
    print("Calculating Third  "+str(iter_i))
    for iter_T in range(0, len(T_set)):
        temp_T = int(T_set[iter_T])
        regretUCB_T[iter_i, iter_T] = UCB1(mu, K, temp_T) / temp_T
        regretMinimax_T[iter_i, iter_T] = \
            BASEFunc(mu, K, temp_T, M, 'minimax', gamma) / temp_T
        regretGeometric_T[iter_i, iter_T] = \
            BASEFunc(mu, K, temp_T, M, 'geometric', gamma) / temp_T
        regretArithmetic_T[iter_i, iter_T] = \
            BASEFunc(mu, K, temp_T, M, 'arithmetic', gamma) / temp_T


regretMinimax_T_mean = np.mean(regretMinimax_T, axis=0)
regretGeometric_T_mean = np.mean(regretGeometric_T, axis=0)
regretArithmetic_T_mean = np.mean(regretArithmetic_T, axis=0)
regretUCB_T_mean = np.mean(regretUCB_T, axis=0)


# comparison with [PRCS16]

regretMinimax = np.zeros((m, len(M_set)), dtype=float)
regretGeometric = np.zeros((m, len(M_set)), dtype=float)
regretPRCSminimax = np.zeros((m, len(M_set)), dtype=float)
regretPRCSgeometric = np.zeros((m, len(M_set)), dtype=float)
regretUCB = np.zeros((m,), dtype=float)
mu = np.concatenate(([mu_max], [mu_min]), axis=0)

for iter_i in range(0, m):
```

```
    print("Calculating Fourth "+str(iter_i))
    regretUCB[iter_i] = UCB1(mu, 2, T)
    for iter_M in range(0, len(M_set)):
        temp_M = M_set[iter_M]
        regretMinimax[iter_i, iter_M] = \
            BASEFunc(mu, 2, T, temp_M, 'minimax', gamma)
        regretGeometric[iter_i, iter_M] = \
            BASEFunc(mu, 2, T, temp_M, 'geometric', gamma)
        regretPRCSminimax[iter_i, iter_M] = \
            PRCS_twoarm(mu, temp_M, T, 'minimax')
        regretPRCSgeometric[iter_i, iter_M] = \
            PRCS_twoarm(mu, temp_M, T, 'geometric')

regretMinimax_mean = np.mean(regretMinimax, axis=0) / T
regretGeometric_mean = np.mean(regretGeometric, axis=0) / T
regretPRCSminimax_mean = np.mean(regretPRCSminimax, axis=0) / T
print("regretPRCSgeometric = \n"+str(regretPRCSgeometric))
regretPRCSGeometric_mean = np.mean(regretPRCSgeometric, axis=0) / T
print("regretPRCSGeometric_mean = \n"+str(regretPRCSGeometric_mean))
regretUCB_mean = np.mean(regretUCB, axis=0) / T


# Figures

plt.figure(0)
plt.plot(M_set, regretMinimax_M_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(M_set, regretGeometric_M_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(M_set, regretArithmetic_M_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(M_set, regretUCB_M_mean * np.ones(len(M_set)), marker='.',
         markerfacecolor='k', linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 8))
plt.xlabel("$M$")
plt.ylabel('Average regret')
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(1)
plt.plot(K_set, regretMinimax_K_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(K_set, regretGeometric_K_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
```

```
plt.plot(K_set, regretArithmetic_K_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(K_set, regretUCB_K_mean, markerfacecolor='k',
         linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 21))
plt.xlabel("$K$")
plt.ylabel('Average regret')
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(2)
plt.plot(T_set, regretMinimax_T_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(T_set, regretGeometric_T_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(T_set, regretArithmetic_T_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(T_set, regretUCB_T_mean, markerfacecolor='k',
         linestyle=':', color='k', linewidth=2)
plt.xlim([int(5e2), int(5e4)])
plt.xscale('log')
plt.xlabel("$T$")
plt.ylabel('Average regret')
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(3)
plt.plot(M_set, regretMinimax_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(M_set, regretGeometric_mean, marker='s', markerfacecolor='b',
         linestyle='--', color='b', linewidth=2)
plt.plot(M_set, regretPRCSminimax_mean, marker='o', markerfacecolor='r',
         linestyle='-', color='r', linewidth=2)
plt.plot(M_set, regretPRCSGeometric_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(M_set, regretUCB_mean * np.ones(len(M_set)),
         markerfacecolor='k', linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 8))
plt.xlabel("$M$")
plt.ylabel('Average regret')
plt.legend(['BaSE: Minimax Grid', 'BaSE: Geometric Grid', 'ETC: Minimax Grid',
    'ETC: Geometric Grid', 'UCB1'])

plt.show()
```

Frammento di codice 5.5: Codice sorgente dopo *transpiling* in Python, *file*: `main.py`

Codice sorgente dopo *transpiling* in Python, *file*: `BASEFunc.py`:

```
import ctypes
import math
import numpy as np


# function of BASE
# parameters
#     K: number of batches
#     TSeq: horizon
#     M: number of batches
#     TGridMinimax: minimax batch grids
#     mu: batch mean
#     gamma: tunning parameter


def BASEFunc(mu, K, T, M, gridType, gamma) -> float:
    # record
    regret = 0
    if gridType == 'minimax':
        a = T ** (1 / (2 - 2**(1 - M)))
        TGrid = np.floor(np.power(
            a, np.subtract(2, np.divide(1, np.power(2, (range(0, M)))))))
        TGrid[M - 1] = T
        # minimax batch
        TGrid = np.concatenate(([0], TGrid), axis=0)
    elif gridType == 'geometric':
        b = T ** (1 / M)
        TGrid = np.floor(np.power(b, range(1, M+1)))
        TGrid[M - 1] = T
        # adaptive batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)
    else:
        TGrid = np.floor(np.linspace(0, T, M + 1))

    # initialization
    activeSet = np.ones((K, 1), dtype=int)
    numberPull = np.zeros(K, dtype=int)
    averageReward = np.zeros(K, dtype=float)

    for i in range(1, M + 1):
        availableK = np.sum(activeSet)
        pullNumber = \
```

```
            int(np.round(np.maximum(np.floor((TGrid[i] -
                                              TGrid[i - 1]) / availableK), 1)))
        TGrid[i] = availableK * pullNumber + TGrid[i - 1]
        rowActiveSetOne, colActiveSetOne = np.where(activeSet == 1)
        for j in rowActiveSetOne:
            averageReward[j] = averageReward[j] * (
                    numberPull[j] / (numberPull[j] + pullNumber)) + (
                    np.mean(np.random.randn(1, pullNumber)) + mu[j]) * (
                    pullNumber / (numberPull[j] + pullNumber))
            regret = regret + (pullNumber * (mu[0] - mu[j]))
            numberPull[j] = numberPull[j] + pullNumber

        rowActiveSetOne, colActiveSetOne = np.where(activeSet == 1)
        maxArm = np.max(averageReward[rowActiveSetOne])
        for j in rowActiveSetOne:
            if (maxArm - averageReward[j]) >= np.sqrt(
                    gamma * math.log(T * K) / numberPull[j]):
                activeSet[j] = 0

    return regret
```

Frammento di codice 5.6: Codice sorgente dopo *transpiling* in Python, *file*: `BASEFunc.py`

Codice sorgente dopo *transpiling* in Python, *file*: `UCB1.py`:

```
import math
import numpy as np


def UCB1(mu, K, T) -> float:
    pullnumber = np.ones(K, dtype=int)
    averageReward = np.zeros(K, dtype=float)

    for t in range(0, K):
        averageReward[t] = mu[t] + np.random.randn()

    for t in range(K, T):
        UCB = averageReward +\
              np.sqrt(2 * np.divide(math.log(T), pullnumber))
        pos = UCB.argmax(0)
        weight = 1 / (pullnumber[pos] + 1)
        averageReward[pos] = (1 - weight) * averageReward[pos] + \
                             weight * (mu[pos] + np.random.randn())
        pullnumber[pos] = pullnumber[pos] + 1
```

```
    regret = (mu[0] - mu[1:]).dot(pullnumber[1:])
    return regret
```

Frammento di codice 5.7: Codice sorgente dopo *transpiling* in Python, *file*: `UCB1.py`

Codice sorgente dopo *transpiling* in Python, *file*: `PRCS_twoarm.py`:

```
import math
import numpy as np


def PRCS_twoarm(mu, M, T, gridType) -> float:
    if gridType == 'minimax':
        a = T**(1 / (2 - 2**(1-M)))
        TGrid = np.floor(np.power(a, np.subtract(2, np.divide(
            1, np.power(2, (range(0, M)))))))
        TGrid[M-1] = T
        # minimax batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)
    elif gridType == 'geometric':
        b = T**(1 / M)
        TGrid = np.floor(np.power(b, range(1, M+1)))
        TGrid[M-1] = T
        # adaptive batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)

    pullnumber = int(np.round(TGrid[1] / 2))
    regret = pullnumber * (mu[0] - mu[1])
    reward = np.sum(np.concatenate((
        [np.add(np.random.randn(pullnumber,), mu[0])],
        [np.add(np.random.randn(pullnumber,), mu[1])]), axis=0), axis=1)
    opt = 0

    for m in range(1, M):
        t = TGrid[m]
        thres = np.sqrt(4 * math.log(2 * T / t) / t)
        if opt == 0:
            if (reward[0] - reward[1]) / t > thres:
                opt = 1
            elif (reward[1] - reward[0]) / t > thres:
                opt = 2
            else:
                cur_number = int(np.round((TGrid[m+1] - TGrid[m]) / 2))
                pullnumber = pullnumber + cur_number
                reward = reward + np.sum(np.concatenate((
```

```
                    [np.add(np.random.randn(cur_number,), mu[0])],
                    [np.add(np.random.randn(cur_number,), mu[1])]),
                    axis=0), axis=1)
                regret = regret + cur_number * (mu[0] - mu[1])

        if opt == 2:
            regret = regret + (TGrid[m + 1] - TGrid[m]) * (mu[0] - mu[1])

        if m == (M-2):
            if reward[0] > reward[1]:
                opt = 1
            else:
                opt = 2

    return regret
```

Frammento di codice 5.8: Codice sorgente dopo *transpiling* in Python, *file*: `PRCS_twoarm.py`

5.2.3 Risultati, confronti e osservazioni

I risultati degli esperimenti sia in Matlab (codice fornito dagli autori) che in Python (post-*transpiling*) sono coerenti qualitativamente e quantitativamente simili. Questi risultati se confrontati con i risultati empirici degli autori risultano pressochè identici in tre su quattro casi. Riepilogando:

- Figura 5.6: *regret* medio vs. il numero dei *batches* M: Le 4 linee sono quasi identiche. Piccola variazione in alto per la *geometric grid* di BaSE nel valore del *regret* medio quando si hanno $M = 6$ *batches*.
- Figura 5.7: *regret* medio vs. il numero degli *arms* K: Le 4 linee sono quasi identiche.
- Figura 5.8: *regret* medio vs. l'orizzonte temporale T: Le 4 linee sono quasi identiche. Piccola variazione in alto per la *geometric grid* e la *arithmetic grid* di BaSE nel valore del *regret* medio quando si hanno piccoli orizzonti temporali, ovvero pochi dati.
- Figura 5.9: *regret* medio vs. il numero dei *batches* M come confronto tra BaSE vs. ETC vs. *UCB1*): 3 della 4 linee sono quasi identiche. La linea indicante il *regret* medio per la *geometric grid* di ETC è molto diversa da quella gli autori hanno fornito nel loro *paper*.

Dalle figure si nota che i risultati degli autori coincidono con quelli delle prove per 3 dei 4 esperimenti. Nell'ultimo esperimento solo 1 delle 4 linee è diversa dai risultati ottenuti dagli autori, con lo stesso codice eseguito, con gli stessi parametri. Esaminando il codice per il calcolo del *regret* medio per la *geometric grid* di ETC risulta sia concorde con l'algoritmo proposto da Perchet, Rigollet, Chassang e Snowberg nel *paper* «Batched bandit problems»[4]. Se le differenze dovessero

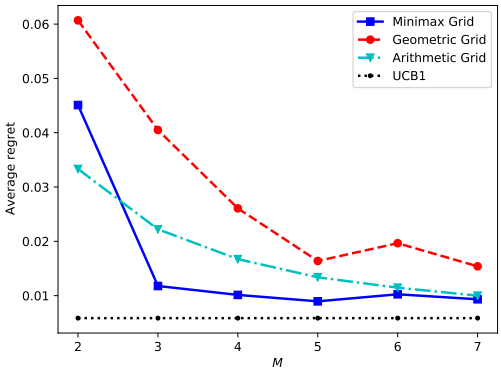


Figura 5.6: *Regret* medio vs. numero di *batches* M.

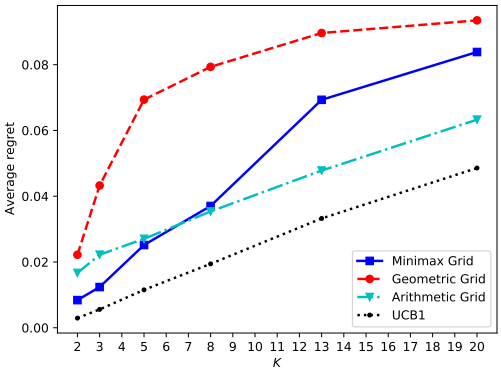


Figura 5.7: *Regret* medio vs. numero di *arms* K.

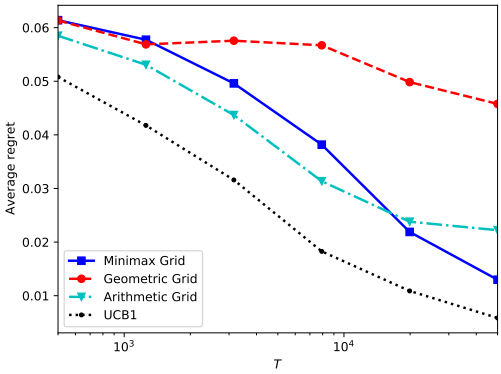


Figura 5.8: *Regret* medio vs. orizzonte temporale T.

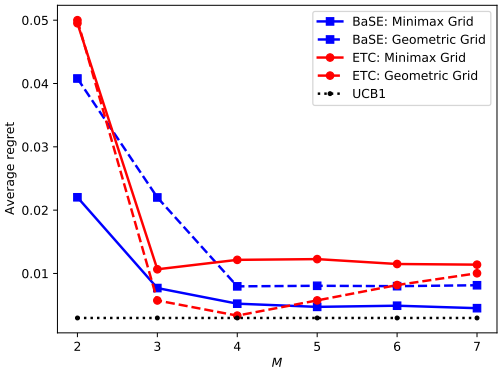


Figura 5.9: Confronto tra BaSE e ETC.

Figura 5.10: Risultati empirici delle *regret performance* degli esperimenti della BaSE *policy* con dati *random*.

confermarsi anche da altri *reviewers*, significherebbe che la BaSE *policy* è meno performante di quanto sostenuto nei casi i *batches* siano nell'intervallo da 3 a 6. Gli autori dello studio sono stati messi a conoscenza di questa piccola differenza nei risultati ma non hanno ancora risposto.

5.3 Nuovi esperimenti su *dataset* con dati reali

Gli stessi esperimenti effettuati dagli autori dello studio sono stati effettuati di nuovo ma ora con dati ottenuti da un *dataset* reale. Il cambiamento da dati generati a *random* a dati provenienti da *dataset* reali ha richiesto delle modifiche sostanziali del codice. Facendo uso della libreria Pandas, prima di inziare con i calcoli, si è fatta una prima preparazione e *cleanup* dei dati. Successivamente, sono stati eseguiti gli esperimenti con l'unico parametro modificato $m = 100$ invece che 200. Questa scelta viene motivata nella sottosezione 5.3.2 - Parametri e considerazioni sul metodo.

5.3.1 Dati e *dataset*

Come menzionato sopra, i dati sono stati ottenuto da un *dataset* aperto, pubblico *online*. Nel percorso di ricerca dei *dataset* adatti all'argomento, si è fatto uso del Forum del sito *web* `https://www.kaggle.com/`[40]. Dalle discussioni nel forum e la ricerca di *datasets* adatti a esperimenti di BMaB, si è "catapultati" al sito di `http://www.grouplens.org/`[41] e specificatamente a `https://grouplens.org/datasets/movielens/`[42]. Tra i diversi *dataset* valutati, si è scelto di usare la *MovieLens 1M movie ratings* MovieLens (2003)[43][44] durante le modifiche del codice e il testing del nuovo programma.

[40] Alphabet Inc. *Kaggle website*. Inglese. Kaggle, una susidiary di Google LLC, è una community online di data scientist e professionisti del machine learning. Kaggle consente agli utenti di trovare e pubblicare datasets, esplorare e costruire modelli in un ambiente di data science basato sul web, lavorare con altri data scientist e ingegneri del machine learning e partecipare a concorsi per risolvere data science challenges. 2020. URL: `https://www.kaggle.com/`.

[41] GroupLens Research. *GroupLens website*. Inglese. GroupLens Research è un laboratorio di ricerca sull'interazione human–computer nel Dipartimento di Informatica e Ingegneria dell'Università del Minnesota. 2020. URL: `http://www.grouplens.org/`.

[42] GroupLens Research. *MovieLens website data*. Inglese. GroupLens Research ha raccolto e reso disponibili datasets del sito web MovieLens. 2020. URL: `https://grouplens.org/datasets/movielens/`.

[43] MovieLens. *MovieLens 1M movie ratings*. Inglese. Stable benchmark dataset. 1 million ratings from 6000 users on 4000 movies. Feb. 2003. URL: `https://grouplens.org/datasets/movielens/1m/`.

[44] F. Maxwell Harper e Joseph A. Konstan. «The MovieLens Datasets: History and Context». Inglese. In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (dic. 2015). Documento locale in pdf, pagine 235–256. ISSN: 2160-6455. DOI: `10.1145/2827872`.

Il *dataset MovieLens 1M movie ratings* è composto di 4 *file* che contengono 1000209 voti anonimi di circa 3900 film realizzato da 6040 utenti di *MovieLens* che si sono uniti a *MovieLens* nel 2000. Tutti i voti sono contenuti nel *file* "rating.dat" con formato: UserID::MovieID::Rating::Timestamp . Di questi dati vengono usati *UserID* come identificatore di *bandits*, *MovieID* come campione di una tirata di un *arm*. Viene usato *Rating* come il *reward* per quella tirata e il *Timestamp* per l'orizzonte temporale. Le informazioni sul film sono nel *file* "movies.dat" con formato: MovieID::Title::Genres . Di questi dati vengono usati *MovieID* come identificatore di *bandits* e *Genres* come *arms* che i *bandits* possono tirare. Le informazioni sugli utenti sono nel *file* "users.dat" con formato: UserID::Gender::Age::Occupation::Zip-code . Questi dati non vengono usati. L'ultimo *file* è "README" dove sono indicate le istruzioni d'uso, la licenza e altre informazioni sui dati.

Una volta finite le modifiche e il testing del codice si è passati a eseguire gli esperimenti su un *dataset* più consistente. Si è scelto di usare *MovieLens 10M movie ratings* MovieLens (2009)[45][44]. Questo *dataset* contiene 10000054 voti e 95580 tag applicati a 10681 film da 71567 utenti del servizio di raccomandazione film *online MovieLens*. Gli utenti sono stati selezionati a caso. Tutti gli utenti selezionati hanno valutato almeno 20 film. A differenza dei precedenti set di dati *MovieLens*, non sono incluse informazioni demografiche. Ogni utente è rappresentato da un ID e non vengono fornite altre informazioni. Il *dataset* è composto di sei *file*. I dati sono contenuti in tre di essi, "movies.dat", "rating.dat" e "tags.dat". Sono inclusi anche gli script per la generazione di sottoinsiemi di dati a supporto di cinque convalide incrociate delle previsioni di rating. La struttura dei dati di "movies.dat" e "rating.dat" è uguale a quella dei *file* nel *dataset MovieLens 1M movie ratings*.

Il secondo *dataset*, avendo più dati di *ratings* (*rewards*) da più *users* (*bandits*) su più *movies* (singole tirate di *glsplarm*), permette di avere un tasso di casualità maggiore. Il numero di *genres* (*arms*) rimane comunque 18, che è minore del K maggiore del secondo esperimento ($K = 20$), altra piccola differenza che però non si rivelerà qualitativamente molto influente.

5.3.2 Parametri e considerazioni sul metodo

Tanti parametri sono uguali a quelli usati dagli autori dello studio. Cambiano il numero dei *bandits* presi in considerazione, da $m = 200$ nello studio originale a $m = 100$. Questo perché nonostante il numero degli *users* (*bandits*) sia elevato nel *dataset*, la gran maggioranza di essi ha recensito (fornito un *reward per* tanti film appartenenti a *genres* (*arms*) loro preferiti, ma non a

[45] MovieLens. *MovieLens 10M movie ratings*. Inglese. Stable benchmark dataset. 10 million ratings and 100000 tag applications applied to 10000 movies by 72000 users. Gen. 2009. URL: https://grouplens.org/datasets/movielens/10m/.

tutti e 18 i *genres*. Quindi si è optato di scegliere i *top* 100 *users* (*bandits*) che avessero recensito (fornito un *reward per* almeno T film.

Il *reward* è $\mu^\star = 2$ per l'*arm* ottimale, ovvero per i film recensiti con voto 4 o 5. Il *reward* è $\mu = 1$ per tutti gli altri *arms*, ovvero per i film recensiti con voti da 1 a 3.

Come menzionato prima altra modifica di parametro è l'intervallo di K nel secondo esperimente che non va più da 2 a 20 ma da 2 a 18. Un problema che condiziona il risultato è il fatto che gli utenti oltre ai generi loro preferiti, tendono a recensire pochi film di altri generi. Questo comporta pochi dati per K oltre il 4 su cui fare dei calcoli.

Un problema simile ma leggermente meno influenzante si nota nel terzo esperimento, dove viene stimato e poi confrontato il *regret* medio vs. l'orizzonte temporale T. Al crescere di T, quindi di tirate da parte dei *bandits*, mantenedo $K = 3$ fisso, diminuisce il numero di *bandits* che hanno recensito decine di migliaia di film, tanto da rendere non sensato $m = 100$.

5.3.3 Codice Python usabile su *dataset*

Il programma in Python degli esperimenti ha subito modifiche per permettere l'importazione dei dati, la preparazione e l'uso all'interno delle funzioni di calcolo secondo le *policies*. Il codice in Python post-*transpiling* e successivamente post-modifiche per lavorare su *dataset* contenenti dati reali, è questo:

Codice sorgente in Python modificato per uso su dati reali, *file*: `main_data.py`:

```
import math
import numpy as np
import pandas as pd

from matplotlib import pyplot as plt

from prepare_data import prepare_data
from UCB1_data import UCB1_data
from BASEFunc_data import BASEFunc_data
from PRCS_twoarm_data import PRCS_twoarm_data

# Read dataset from file
print('Inizio lettura dataset film...')
movies = pd.read_csv("../../Datasets/ml-10M100K/movies.dat", sep='[:][:]',
    engine='python', header=None)
print('Fine lettura dataset film.')
print('Inizio lettura dataset recensioni...')
ratings = pd.read_csv("../../Datasets/ml-10M100K/ratings.dat", sep='[:][:]',
    engine='python', header=None)
```

```
print('Fine lettura dataset recensioni.')

"""
Parametri
"""

# K = 3 # Number of genres (arms) that the chosen user (bandit) has rated the
    most
# M = 3  # Numero dei batches
# m = 100
K = 3
M = 3
m = 200  # Users/Bandits = 200

# Prepare data
ratingsOfChosenBanditWithGenres = prepare_data(movies, ratings, m)

# K_set = vettore riga di 6 punti da log10(2) a log10(18) ugualmente
# logaritmicamente distanti (arrotondati). 18 e non 120 perche gli arms
# del dataset sono 18
# M_set = insieme dei punti da 2 a 7 (compreso).
K_set = np.round(np.logspace(math.log(2, 10),
                             math.log(18, 10),
                             num=6,
                             endpoint=True,
                             base=10.0,
                             dtype=None, axis=0))
M_set = range(2, 8)

# mu_max = 2 reward medio per l'arm ottimale
# gamma = 1
mu_max = 2  # Max rating/reward of movies/arm pull = 1
mu_min = 1  # Min rating/reward of movies/arm pull = -1
gamma = 1

"""
Esperimenti
"""

# dipendenza da M

# regretMinimax_M = matrice di zeri di dimensione mxlen(M_set)
# regretGeometric_M = matrice di zeri di dimensione mxlen(M_set)
```

```
# regretArithmetic_M = matrice di zeri di dimensione mxlen(M_set)
# regretUCB_M = vettore di zeri di dimensione mx1
# mu = i rewards come concatenazione (bind di colonne) dei
# reward di mu_max e tutti i K-1 mu_min
regretMinimax_M = np.zeros((m, len(M_set)), dtype=float)
regretGeometric_M = np.zeros((m, len(M_set)), dtype=float)
regretArithmetic_M = np.zeros((m, len(M_set)), dtype=float)
regretUCB_M = np.zeros((m,), dtype=float)
mu = np.concatenate(([mu_max], mu_min * np.ones((1, K-1),
                                                dtype=int).flatten()), axis=0)

# T = numero di pulls nel time orizon
T = 0
T_allVariations = []


for iter_i in range(0, m):
    print("Calcolando Diagramma 1, iter_i(da 0 a ", str(m - 1), ") =", str(
    iter_i))

    # Choosing the only the ratings of movies from the ith bandit
    ratingsOfChosenBanditWithGenresith = \
        ratingsOfChosenBanditWithGenres[ratingsOfChosenBanditWithGenres[
            'Bandit'].isin(ratingsOfChosenBanditWithGenres['Bandit'].
    value_counts()[iter_i:iter_i+1].index.tolist())]

    leastLongTimeHorizon = 0
    ratingsOfChosenBanditWithGenresithdata = ratingsOfChosenBanditWithGenresith

    while leastLongTimeHorizon < 2:
        # Choosing the ratings on the k arms/genres most rated from the ith
    bandit
        ratingsOfChosenBanditWithGenresithdata =
    ratingsOfChosenBanditWithGenresith[
            ratingsOfChosenBanditWithGenresith[
                'Arm_K'].isin(ratingsOfChosenBanditWithGenresith['Arm_K'].
    value_counts()[:K].index.tolist())]

        leastLongTimeHorizon = ratingsOfChosenBanditWithGenresithdata['Arm_K'].
    value_counts().min()

        # Dropping di rating rows per genre/arm oltre il leastLongTimeHorizon
        # Renumbering of the arms from 1 to K
```

```
        # Numbering time horizin
        newData = []
        k = 1
        for arm in ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique().
    tolist():
            newDataFrame = ratingsOfChosenBanditWithGenresithdata[
                ratingsOfChosenBanditWithGenresithdata['Arm_K'] == arm].head(
    leastLongTimeHorizon)
            newDataFrame['Arm_K'] = k
            k = k + 1
            newDataFrame['Time_T'] = range(1, leastLongTimeHorizon + 1)
            newData.append(newDataFrame)

        ratingsOfChosenBanditWithGenresithdata = pd.concat(newData)

        if len(ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique().tolist
    ()) < K:
            K = K - 1
        else:
            break

    T = leastLongTimeHorizon
    T_allVariations.append(T)

    regretUCB_M[iter_i] = UCB1_data(ratingsOfChosenBanditWithGenresithdata, mu,
     K, T)
    for iter_M in range(0, len(M_set)):
        temp_M = M_set[iter_M]
        regretMinimax_M[iter_i, iter_M] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, T,
    temp_M, 'minimax', gamma)
        regretGeometric_M[iter_i, iter_M] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, T,
    temp_M, 'geometric', gamma)
        regretArithmetic_M[iter_i, iter_M] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, T,
    temp_M, 'arithmetic', gamma)

T_first_min = min(T_allVariations)
T_first_max = max(T_allVariations)
T = np.mean(T_allVariations, axis=0)
T_first = T
```

```
regretMinimax_M_mean = np.mean(regretMinimax_M, axis=0) / T
regretGeometric_M_mean = np.mean(regretGeometric_M, axis=0) / T
regretArithmetic_M_mean = np.mean(regretArithmetic_M, axis=0) / T
regretUCB_M_mean = np.mean(regretUCB_M, axis=0) / T

# dependence on K

regretMinimax_K = np.zeros((m, len(K_set)), dtype=float)
regretGeometric_K = np.zeros((m, len(K_set)), dtype=float)
regretArithmetic_K = np.zeros((m, len(K_set)), dtype=float)
regretUCB_K = np.zeros((m, len(K_set)), dtype=float)

T_allVariations.clear()
for iter_i in range(0, m):
    print("Calcolando Diagramma 2, iter_i(da 0 a ", str(m - 1), ") =", str(
    iter_i))

    # Choosing the only the ratings of movies from the ith bandit
    ratingsOfChosenBanditWithGenresith = \
        ratingsOfChosenBanditWithGenres[ratingsOfChosenBanditWithGenres[
            'Bandit'].isin(ratingsOfChosenBanditWithGenres['Bandit'].
    value_counts()[iter_i:iter_i+1].index.tolist())]

    for iter_K in range(0, len(K_set)):
        temp_K = int(K_set[iter_K])
        leastLongTimeHorizon = 0
        while 1:
            # Choosing the ratings on the k arms/genres most rated from the ith
     bandit
            ratingsOfChosenBanditWithGenresithdata =
    ratingsOfChosenBanditWithGenresith[
                ratingsOfChosenBanditWithGenresith[
                    'Arm_K'].isin(ratingsOfChosenBanditWithGenresith['Arm_K'].
    value_counts()[:temp_K].index.tolist())]

            leastLongTimeHorizon = ratingsOfChosenBanditWithGenresithdata['
    Arm_K'].value_counts().min()

            # Dropping di rating rows per genre/arm oltre il
    leastLongTimeHorizon
            # Renumbering of the arms from 1 to K
            # Numbering time horizin
            newData = []
```

```
            k = 1
            for arm in ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique()
    .tolist():
                newDataFrame = ratingsOfChosenBanditWithGenresithdata[
                    ratingsOfChosenBanditWithGenresithdata['Arm_K'] == arm].
    head(leastLongTimeHorizon)
                newDataFrame['Arm_K'] = k
                k = k + 1
                newDataFrame['Time_T'] = range(1, leastLongTimeHorizon + 1)
                newData.append(newDataFrame)

            ratingsOfChosenBanditWithGenresithdata = pd.concat(newData)

            if len(ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique().
    tolist()) < temp_K:
                temp_K = temp_K - 1
            else:
                break

        T = leastLongTimeHorizon
        T_allVariations.append(T)

        mu = np.concatenate(([mu_max], mu_min *
                             np.ones((1, temp_K - 1),
                                     dtype=int).flatten()), axis=0)

        regretUCB_K[iter_i, iter_K] = UCB1_data(
    ratingsOfChosenBanditWithGenresithdata, mu, temp_K, T)
        regretMinimax_K[iter_i, iter_K] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, temp_K, T
    , M, 'minimax', gamma)
        regretGeometric_K[iter_i, iter_K] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, temp_K, T
    , M, 'geometric', gamma)
        regretArithmetic_K[iter_i, iter_K] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, temp_K, T
    , M, 'arithmetic', gamma)

T = np.mean(T_allVariations, axis=0)

regretMinimax_K_mean = np.mean(regretMinimax_K, axis=0) / T
regretGeometric_K_mean = np.mean(regretGeometric_K, axis=0) / T
regretArithmetic_K_mean = np.mean(regretArithmetic_K, axis=0) / T
```

```
regretUCB_K_mean = np.mean(regretUCB_K, axis=0) / T

# dependence on T

K = 3
T = T_first

# logaritmicamente distanti (arrotondati)
T_set = np.round(np.logspace(math.log(T_first_min, 10), math.log(T_first_max,
    10), num=6,
                             endpoint=True,
                             base=10.0,
                             dtype=None, axis=0))

regretMinimax_T = np.zeros((m, len(T_set)), dtype=float)
regretGeometric_T = np.zeros((m, len(T_set)), dtype=float)
regretArithmetic_T = np.zeros((m, len(T_set)), dtype=float)
regretUCB_T = np.zeros((m, len(T_set)), dtype=float)
mu = np.concatenate(([mu_max], mu_min * np.ones((1, K - 1),
                                                dtype=int).flatten()), axis=0)

for iter_i in range(0, m):
    print("Calcolando Diagramma 3, iter_i(da 0 a ", str(m - 1), ") =", str(
    iter_i))

    # Choosing the only the ratings of movies from the ith bandit
    ratingsOfChosenBanditWithGenresith = \
        ratingsOfChosenBanditWithGenres[ratingsOfChosenBanditWithGenres[
            'Bandit'].isin(ratingsOfChosenBanditWithGenres['Bandit'].
    value_counts()[iter_i:iter_i + 1].index.tolist())]

    for iter_T in range(0, len(T_set)):
        leastLongTimeHorizon = 0
        ratingsOfChosenBanditWithGenresithdata =
    ratingsOfChosenBanditWithGenresith

        while leastLongTimeHorizon < 2:
            # Choosing the ratings on the k arms/genres most rated from the ith
     bandit
            ratingsOfChosenBanditWithGenresithdata =
    ratingsOfChosenBanditWithGenresith[
                ratingsOfChosenBanditWithGenresith[
                    'Arm_K'].isin(ratingsOfChosenBanditWithGenresith['Arm_K'].
```

```
value_counts()[:K].index.tolist())]

        leastLongTimeHorizon = ratingsOfChosenBanditWithGenresithdata['
Arm_K'].value_counts().min()

        # Dropping di rating rows per genre/arm oltre il
leastLongTimeHorizon
        # Renumbering of the arms from 1 to K
        # Numbering time horizin
        newData = []
        k = 1
        for arm in ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique()
.tolist():
            newDataFrame = ratingsOfChosenBanditWithGenresithdata[
                ratingsOfChosenBanditWithGenresithdata['Arm_K'] == arm].
head(leastLongTimeHorizon)
            newDataFrame['Arm_K'] = k
            k = k + 1
            newDataFrame['Time_T'] = range(1, leastLongTimeHorizon + 1)
            newData.append(newDataFrame)

        ratingsOfChosenBanditWithGenresithdata = pd.concat(newData)

        if len(ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique().
tolist()) < K:
            K = K - 1
        else:
            break

    temp_T = np.minimum(leastLongTimeHorizon, int(T_set[iter_T]))
    T_allVariations.append(temp_T)

    regretUCB_T[iter_i, iter_T] = UCB1_data(
ratingsOfChosenBanditWithGenresithdata, mu, K, temp_T) / temp_T
    regretMinimax_T[iter_i, iter_T] = \
        BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, temp_T
, M, 'minimax', gamma) / temp_T
    regretGeometric_T[iter_i, iter_T] = \
        BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, temp_T
, M, 'geometric', gamma) / temp_T
    regretArithmetic_T[iter_i, iter_T] = \
        BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, K, temp_T
, M, 'arithmetic', gamma) / temp_T
```

```
regretMinimax_T_mean = np.mean(regretMinimax_T, axis=0)
regretGeometric_T_mean = np.mean(regretGeometric_T, axis=0)
regretArithmetic_T_mean = np.mean(regretArithmetic_T, axis=0)
regretUCB_T_mean = np.mean(regretUCB_T, axis=0)

# comparison with [PRCS16]

regretMinimax = np.zeros((m, len(M_set)), dtype=float)
regretGeometric = np.zeros((m, len(M_set)), dtype=float)
regretPRCSminimax = np.zeros((m, len(M_set)), dtype=float)
regretPRCSgeometric = np.zeros((m, len(M_set)), dtype=float)
regretUCB = np.zeros((m,), dtype=float)
mu = np.concatenate(([mu_max], [mu_min]), axis=0)

T_allVariations.clear()
for iter_i in range(0, m):
    print("Calcolando Diagramma 4, iter_i(da 0 a ", str(m - 1), ") =", str(
    iter_i))

    # Choosing the only the ratings of movies from the ith bandit
    ratingsOfChosenBanditWithGenresith = \
        ratingsOfChosenBanditWithGenres[ratingsOfChosenBanditWithGenres[
            'Bandit'].isin(ratingsOfChosenBanditWithGenres['Bandit'].
    value_counts()[iter_i:iter_i + 1].index.tolist())]

    leastLongTimeHorizon = 0
    ratingsOfChosenBanditWithGenresithdata = ratingsOfChosenBanditWithGenresith

    while leastLongTimeHorizon < 2:
        # Choosing the ratings on the k arms/genres most rated from the ith
    bandit
        ratingsOfChosenBanditWithGenresithdata =
    ratingsOfChosenBanditWithGenresith[
            ratingsOfChosenBanditWithGenresith[
                'Arm_K'].isin(ratingsOfChosenBanditWithGenresith['Arm_K'].
    value_counts()[:2].index.tolist())]

        leastLongTimeHorizon = ratingsOfChosenBanditWithGenresithdata['Arm_K'].
    value_counts().min()

        # Dropping di rating rows per genre/arm oltre il leastLongTimeHorizon
        # Renumbering of the arms from 1 to K
```

```
        # Numbering time horizin
        newData = []
        k = 1
        for arm in ratingsOfChosenBanditWithGenresithdata['Arm_K'].unique().
    tolist():
            newDataFrame = ratingsOfChosenBanditWithGenresithdata[
                ratingsOfChosenBanditWithGenresithdata['Arm_K'] == arm].head(
    leastLongTimeHorizon)
            newDataFrame['Arm_K'] = k
            k = k + 1
            newDataFrame['Time_T'] = range(1, leastLongTimeHorizon + 1)
            newData.append(newDataFrame)

        ratingsOfChosenBanditWithGenresithdata = pd.concat(newData)

    T = leastLongTimeHorizon
    T_allVariations.append(T)

    regretUCB[iter_i] = UCB1_data(ratingsOfChosenBanditWithGenresithdata, mu,
    2, T)
    for iter_M in range(0, len(M_set)):
        temp_M = M_set[iter_M]
        regretMinimax[iter_i, iter_M] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, 2, T,
    temp_M, 'minimax', gamma)
        regretGeometric[iter_i, iter_M] = \
            BASEFunc_data(ratingsOfChosenBanditWithGenresithdata, mu, 2, T,
    temp_M, 'geometric', gamma)
        regretPRCSminimax[iter_i, iter_M] = \
            PRCS_twoarm_data(ratingsOfChosenBanditWithGenresithdata, mu, temp_M
    , T, 'minimax')
        regretPRCSgeometric[iter_i, iter_M] = \
            PRCS_twoarm_data(ratingsOfChosenBanditWithGenresithdata, mu, temp_M
    , T, 'geometric')

T = np.mean(T_allVariations, axis=0)

regretMinimax_mean = np.mean(regretMinimax, axis=0) / T
regretGeometric_mean = np.mean(regretGeometric, axis=0) / T
regretPRCSminimax_mean = np.mean(regretPRCSminimax, axis=0) / T
regretPRCSGeometric_mean = np.mean(regretPRCSgeometric, axis=0) / T
regretUCB_mean = np.mean(regretUCB, axis=0) / T
```

```
# Figures

plt.figure(0)
plt.plot(M_set, regretMinimax_M_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(M_set, regretGeometric_M_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(M_set, regretArithmetic_M_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(M_set, regretUCB_M_mean * np.ones(len(M_set)), marker='.',
         markerfacecolor='k', linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 8))
plt.xlabel("M")
plt.ylabel('Average regret')
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(1)
plt.plot(K_set, regretMinimax_K_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(K_set, regretGeometric_K_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(K_set, regretArithmetic_K_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(K_set, regretUCB_K_mean, markerfacecolor='k',
         linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 19))
plt.xlabel("K")
plt.ylabel('Average regret')
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(2)
plt.plot(T_set, regretMinimax_T_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(T_set, regretGeometric_T_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(T_set, regretArithmetic_T_mean, marker='v', markerfacecolor='c',
         linestyle='-.', color='c', linewidth=2)
plt.plot(T_set, regretUCB_T_mean, markerfacecolor='k',
         linestyle=':', color='k', linewidth=2)
plt.xlim([int(T_first_min), int(T_first_max)])
plt.xscale('log')
plt.xlabel("T")
plt.ylabel('Average regret')
```

```
plt.legend(['Minimax Grid', 'Geometric Grid', 'Arithmetic Grid', 'UCB1'])

plt.figure(3)
plt.plot(M_set, regretMinimax_mean, marker='s', markerfacecolor='b',
         linestyle='-', color='b', linewidth=2)
plt.plot(M_set, regretGeometric_mean, marker='s', markerfacecolor='b',
         linestyle='--', color='b', linewidth=2)
plt.plot(M_set, regretPRCSminimax_mean, marker='o', markerfacecolor='r',
         linestyle='-', color='r', linewidth=2)
plt.plot(M_set, regretPRCSGeometric_mean, marker='o', markerfacecolor='r',
         linestyle='--', color='r', linewidth=2)
plt.plot(M_set, regretUCB_mean * np.ones(len(M_set)),
         markerfacecolor='k', linestyle=':', color='k', linewidth=2)
plt.xticks(range(2, 8))
plt.xlabel("M")
plt.ylabel('Average regret')
plt.legend(['BaSE: Minimax Grid', 'BaSE: Geometric Grid', 'ETC: Minimax Grid',
    'ETC: Geometric Grid', 'UCB1'])

plt.show()
```

Frammento di codice 5.9: Codice sorgente in Python modificato per uso su dati reali, *file*: `main_data.py`

Codice sorgente in Python modificato per uso su dati reali, *file*: `prepare_data.py`:

```
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'


def prepare_data(movies,
                 # users,
                 ratings,
                 m):
    print('Inizio preparazione dati...')
    movies.columns = ['MovieID', 'Title', 'Genres']
    # users.columns = ['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code']
    ratings.columns = ['UserID', 'MovieID', 'Rating', 'Timestamp']

    # Remove second genre from movies categories with multiple genres
    movies['Genres'] = movies['Genres'].str.split('|').str[0]

    print('Inizio sceta top m bandits...')
    # Choosing the bandits the m users with most ratings of movies
```

```
    topmBanditsRatings = ratings[ratings['UserID'].isin(ratings['UserID'].
value_counts()[:m].index.tolist())]
    print('Fine sceta top m bandits.')

    print('Inizio rinomina bandits ids...')
    topmBanditsRatings['UserID'].replace(topmBanditsRatings['UserID'].unique().
tolist(), range(1, m + 1),
                                         inplace=True)
    print('Fine rinomina bandits ids.')

    print('Inizio merging ratings con movie genres...')
    # Merging ratings with movie genras (arms)
    topmBanditsRatings = pd.merge(left=topmBanditsRatings, right=movies,
                                                  left_on='MovieID', right_on
='MovieID').drop(columns=['Title'])
    print('Fine merging bandit ratings con movie genres.')

    print('Inizio rinomina genres...')
    # Map Genres to numbers going from 1 to 18,
    topmBanditsRatings['Genres'].replace(['Action', 'Adventure', 'Animation',
                                          'Children\'s', 'Comedy', 'Crime',
                                          'Documentary', 'Drama', 'Fantasy',
                                          'Film - Noir', 'Horror', 'Musical',
                                          'Mystery', 'Romance', 'Sci - Fi',
                                          'Thriller', 'War', 'Western'],
                                         range(1, 19),
                                         inplace=True)
    print('Fine rinomina genres')

    print('Inizio flattening dei ratings...')
    topmBanditsRatings['Rating'].replace(range(1, 6), [1, 1, 1, 2, 2], inplace=
True)
    print('Fine flattening ratings.')

    print('Inizio rimozione e rinomina colonne...')
    # Elimino colonne non necessarie, rinomino quelle necessarie
    topmBanditsRatings = topmBanditsRatings.drop(columns=["MovieID"]).rename(
        columns={'UserID': 'Bandit', 'Rating': 'Reward_mu', 'Timestamp': '
Time_T', 'Genres': 'Arm_K'})
    print('Fine rimozione e rinomina colonne.')

    print('Fine preparazione dati.')
```

```
    return topmBanditsRatings
```

Frammento di codice 5.10: Codice sorgente in Python modificato per uso su dati reali, *file*: `prepare_data.py`

Codice sorgente in Python modificato per uso su dati reali, *file*: `BASEFunc_data.py`:

```
import math
import numpy as np


# function of BASE
# parameters
#     K: number of batches
#     TSeq: horizon
#     M: number of batches
#     TGridMinimax: minimax batch grids
#     mu: batch mean
#     gamma: tunning parameter


def BASEFunc_data(ratingsOfChosenBanditWithGenres, mu, K, T, M, gridType, gamma
    ) -> float:
    # record
    regret = 0
    TGrid = []
    if gridType == 'minimax':
        a = T ** (1 / (2 - 2**(1 - M)))
        TGrid = np.floor(np.power(
            a, np.subtract(2, np.divide(1, np.power(2, (range(0, M)))))))
        TGrid[M - 1] = T
        # minimax batch
        TGrid = np.concatenate(([0], TGrid), axis=0)
    elif gridType == 'geometric':
        b = T ** (1 / M)
        TGrid = np.floor(np.power(b, range(1, M+1)))
        TGrid[M - 1] = T
        # adaptive batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)
    else:
        TGrid = np.floor(np.linspace(0, T, M + 1))

    # initialization
    activeSet = np.ones((K, 1), dtype=int)
```

```
    numberPull = np.zeros(K, dtype=int)
    averageReward = np.zeros(K, dtype=float)

    timeHorizonRead = 1

    for i in range(1, M + 1):
        availableK = np.sum(activeSet)
        pullNumber = np.minimum(int(np.round(np.maximum(np.floor((TGrid[i] -
                                              TGrid[i - 1]) / availableK), 1)))
    , T-timeHorizonRead)
        TGrid[i] = availableK * pullNumber + TGrid[i - 1]
        rowActiveSetOne, colActiveSetOne = np.where(activeSet == 1)
        for j in rowActiveSetOne:
            rewardsOfThisTimeHorizon = ratingsOfChosenBanditWithGenres[
                (ratingsOfChosenBanditWithGenres['Arm_K'] == j + 1) &
                (ratingsOfChosenBanditWithGenres['Time_T'] >= timeHorizonRead)
    &
                (ratingsOfChosenBanditWithGenres['Time_T'] < (timeHorizonRead +
     pullNumber))
                ]['Reward_mu'].to_numpy()
            timeHorizonRead = np.minimum(timeHorizonRead + pullNumber, T)
            averageReward[j] = averageReward[j] * (
                    numberPull[j] / (numberPull[j] + pullNumber)) + (
                    np.mean(rewardsOfThisTimeHorizon) + mu[j]) * (
                    pullNumber / (numberPull[j] + pullNumber))
            regret = regret + (pullNumber * (mu[0] - mu[j]))
            numberPull[j] = numberPull[j] + pullNumber

        rowActiveSetOne, colActiveSetOne = np.where(activeSet == 1)
        maxArm = np.max(averageReward[rowActiveSetOne])
        for j in rowActiveSetOne:
            if (maxArm - averageReward[j]) >= np.sqrt(
                    gamma * math.log(T * K) / numberPull[j]):
                activeSet[j] = 0

    return regret
```

Frammento di codice 5.11: Codice sorgente in Python modificato per uso su dati reali, *file*: `BASEFunc_data.py`

Codice sorgente in Python modificato per uso su dati reali, *file*: `UCB1_data.py`:

```
import math
import numpy as np
```

```
def UCB1_data(ratingsOfChosenBanditWithGenres, mu, K, T) -> float:
    pullnumber = np.ones(K, dtype=int)
    averageReward = np.zeros(K, dtype=float)

    for k in range(0, K):
        rewardsOfFirstTimeHorizon = ratingsOfChosenBanditWithGenres[
            (ratingsOfChosenBanditWithGenres['Arm_K'] == k + 1) &
            (ratingsOfChosenBanditWithGenres['Time_T'] == 1)
            ]['Reward_mu'].to_numpy()
        averageReward[k] = mu[k] + rewardsOfFirstTimeHorizon

    for t in range(1, T):
        UCB = averageReward +\
              np.sqrt(2 * np.divide(math.log(T), pullnumber))
        pos = UCB.argmax(0)
        weight = 1 / (pullnumber[pos] + 1)
        rewardsOfThisTimeHorizon = ratingsOfChosenBanditWithGenres[
            (ratingsOfChosenBanditWithGenres['Arm_K'] == pos + 1) &
            (ratingsOfChosenBanditWithGenres['Time_T'] == t)
            ]['Reward_mu'].to_numpy()
        averageReward[pos] = (1 - weight) * averageReward[pos] + \
                             weight * (mu[pos] + rewardsOfThisTimeHorizon)

        pullnumber[pos] = pullnumber[pos] + 1

    regret = (mu[0] - mu[1:]).dot(pullnumber[1:])
    return regret
```

Frammento di codice 5.12: Codice sorgente in Python modificato per uso su dati reali, *file*: `UCB1_data.py`

Codice sorgente in Python modificato per uso su dati reali, *file*: `PRCS_twoarm_data.py`:

```
import math
import numpy as np


def PRCS_twoarm_data(ratingsOfChosenBanditWithGenres, mu, M, T, gridType) ->
    float:
    if gridType == 'minimax':
        a = T**(1 / (2 - 2**(1-M)))
        TGrid = np.floor(np.power(a, np.subtract(2, np.divide(
```

```
            1, np.power(2, (range(0, M)))))))
        TGrid[M-1] = T
        # minimax batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)
    elif gridType == 'geometric':
        b = T**(1 / M)
        TGrid = np.floor(np.power(b, range(1, M+1)))
        TGrid[M-1] = T
        # adaptive batch grids
        TGrid = np.concatenate(([0], TGrid), axis=0)

    pullnumber = int(np.round(TGrid[1] / 2))
    regret = pullnumber * (mu[0] - mu[1])

    rewardsOfThisTimeHorizon1 = ratingsOfChosenBanditWithGenres[
        (ratingsOfChosenBanditWithGenres['Arm_K'] == 1) &
        (ratingsOfChosenBanditWithGenres['Time_T'] == 1)]['Reward_mu'].to_numpy
    ()

    rewardsOfThisTimeHorizon2 = ratingsOfChosenBanditWithGenres[
        (ratingsOfChosenBanditWithGenres['Arm_K'] == 2) &
        (ratingsOfChosenBanditWithGenres['Time_T'] == 1)]['Reward_mu'].to_numpy
    ()

    reward = np.sum(np.concatenate((
        [np.add(rewardsOfThisTimeHorizon1, mu[0])],
        [np.add(rewardsOfThisTimeHorizon2, mu[1])]), axis=0), axis=1)
    opt = 0

    for m in range(1, M):
        t = TGrid[m]
        thres = np.sqrt(4 * math.log(2 * T / t) / t)
        if opt == 0:
            if (reward[0] - reward[1]) / t > thres:
                opt = 1
            elif (reward[1] - reward[0]) / t > thres:
                opt = 2
            else:
                cur_number = int(np.round((TGrid[m+1] - TGrid[m]) / 2))
                pullnumber = pullnumber + cur_number

                rewardsOfThisTimeHorizon1 = ratingsOfChosenBanditWithGenres[
                    (ratingsOfChosenBanditWithGenres['Arm_K'] == 1) &
```

```
                    (ratingsOfChosenBanditWithGenres['Time_T'] >= cur_number) &
                    (ratingsOfChosenBanditWithGenres['Time_T'] < pullnumber)
                    ]['Reward_mu'].to_numpy()

                rewardsOfThisTimeHorizon2 = ratingsOfChosenBanditWithGenres[
                    (ratingsOfChosenBanditWithGenres['Arm_K'] == 2) &
                    (ratingsOfChosenBanditWithGenres['Time_T'] >= cur_number) &
                    (ratingsOfChosenBanditWithGenres['Time_T'] < pullnumber)
                    ]['Reward_mu'].to_numpy()

                reward = reward + np.sum(np.concatenate((
                    [np.add(rewardsOfThisTimeHorizon1, mu[0])],
                    [np.add(rewardsOfThisTimeHorizon2, mu[1])]),
                    axis=0), axis=1)
                regret = regret + cur_number * (mu[0] - mu[1])

        if opt == 2:
            regret = regret + (TGrid[m + 1] - TGrid[m]) * (mu[0] - mu[1])

        if m == (M-2):
            if reward[0] > reward[1]:
                opt = 1
            else:
                opt = 2

    return regret
```

Frammento di codice 5.13: Codice sorgente in Python modificato per uso su dati reali, *file*: `PRCS_twoarm_data.py`

5.3.4 Risultati e osservazioni

Come si può vedere in Figura 5.15, i risultati degli esperimenti su *dataset* contenente dati reali sono spesso con variazioni rispetto agli esperimenti ideali su dati perfettamente casuali.

In Figura 5.11 le linee di *geometric grid* e di *arithmetic grid* (BaSE) variano leggermente da quelle ottenute in Figura 5.1. Gli ordini di grandezza del *regret* medio sono simili.

In Figura 5.12 la linea di *minimax grid* (BaSE) è totalmente diversa da quella ottenuta in Figura 5.2. Le altre tre linee sono totalmente diverse (e si potrebbe dire anche forse poco sensate) per $K > 4$ *arms*. Come spiegato nella sottosezione 5.3.2 - Parametri e considerazioni sul metodo, questo è dovuto alla specificità dei dati nel *dataset*.

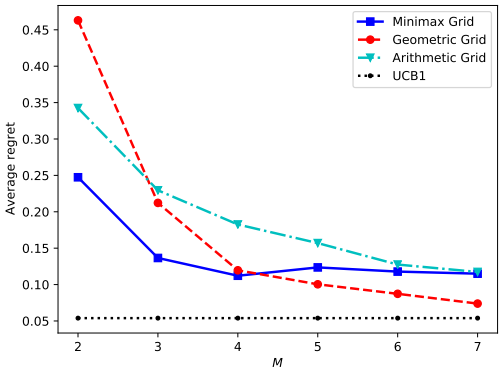


Figura 5.11: *Regret* medio vs. numero di *batches* M.

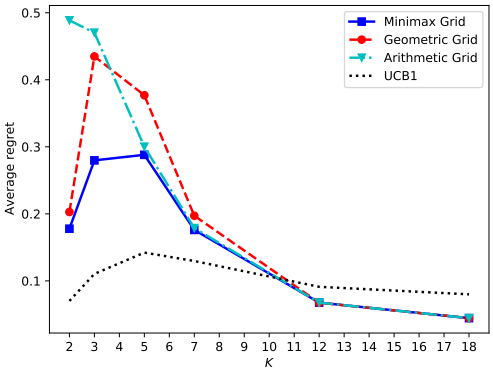


Figura 5.12: *Regret* medio vs. numero di *arms* K.

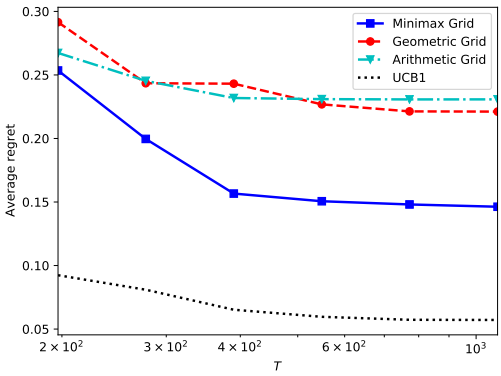


Figura 5.13: *Regret* medio vs. orizzonte temporale T.

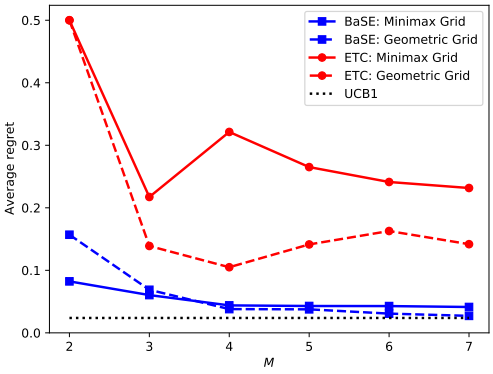


Figura 5.14: Confronto tra BaSE e ETC.

Figura 5.15: Risultati empirici delle *regret performance* degli esperimenti della BaSE *policy* con dati reali.

In Figura 5.13 tutte le linee tendono ad allontanarsi da quelle ottenute in Figura 5.3 per T che crescono. Il motivo, spiegato nella sottosezione 5.3.2 - Parametri e considerazioni sul metodo, è dovuto al numero finito (ordine di pochi milla, ancora meno dentro un genere specifico) film (possibili tiri di un *arm*) recensibili (di cui si dispongono dati di *reward*) da parte degli *users* (*bandits*. L'effetto è evidente anche nell'ordine di grandezza del *regret* medio ottenuto.

La stessa differenza di ordine di grandezza del *regret* medio ottenuto si può notare anche nella Figura 5.14 quando confrontata con Figura 5.4. Le differenze ci sono anche nei singoli confronti delle diverse linee.

Capitolo 6

Conclusioni

Icona del Capitolo 6[2]

6.1 Osservazioni conclusive

Il *paper* fornisce risultati significativi che quantificano l'impatto e l'uso ottimale delle informazioni parziali a causa del *sub-sampling* nei *Multi-armed Bandit* stocastici, un'importante classe di problemi di *online learning*. Si tratta di un'estensione di informazioni parziali nel dominio dello spazio (*feedback* dell'*armed bandit*) al dominio del tempo, quando non è possibile raccogliere un campione di *feedback* in ogni *round*.

Gli autori propongono un nuovo algoritmo, il BaSE. Il contributo tecnico dell'articolo sembra valido. Gli autori hanno analizzato sia il *minimax regret* che il *regret* adattivo con dimensioni di *batches* fisse e adattabili. Hanno dimostrano anche una corrispondente garanzia con limite inferiore, stabilendo ottimalità del metodo proposto. Il documento è nel complesso ben scritto e facile da seguire.

Per le conclusioni sugli esperimenti effettuati, consultare la sottosezione 5.1.3 - Risultati e osservazioni, la sottosezione 5.2.3 - Risultati, confronti e osservazioni e la sottosezione 5.3.4 - Risultati e osservazioni.

Da notare il fatto che quando usato su *dataset* con dati reali, bisogna stare attenti alle particolarità del *dataset*. Molti aspetti potrebbero distrocere i risultati al punto di renderli inutili o peggio ancora portare a conclusioni sbagliate.

6.2 Possibili usi

La *policy Batched Successive Elimination* può trovare uso in tante aree della vita e della scienza. In seguito un elenco di usi in *real-life* e in *machine learning*[46].

6.2.1 Usi nella vita

Alcuni usi possibili in *real-life*:

- **Test clinici**: Ottimizzazione dei trattamenti e dei test clinici su animali o persone limitando il deterioramento della salute del soggetto.
- **Notizie, social media**: Ottimizzazione dei titoli delle notizie per editori e siti multimediali. Un titolo convincente aumenterà i lettori, il coinvolgimento degli utenti e le condivisioni nei social networks.
- **Finanza**: Ottimizzazione e selezione sequenziale di *portfolio* al fine di massimizzare il *reward* cumulativo.
- **eCommerce**: *Pricing* dinamico per prezzi decisi in tempo reale per tanti prodotti, facendo esperimenti sul prezzo con frequenti cambiamenti di esso per conoscere meglio la richiesta e massimizzare i profitti a lungo termine.
- **Sistemi di raccomandazione**: Dilemma del *exploration-exploitation* per fare delle raccomandazioni mirate ad un utente in base a sue preferenze.
- **Altri usi**: *Information retrieval*, *influence maximizationm* o *dialogue systems*. *Anomaly detection* e telecomunicazioni.

6.2.2 Usi in ambito *machine learning* (e alcuni argomenti visti a lezione)

Alcuni usi possibili in ambito *machine learning* (e alcuni di questi argomenti sono stati trattati concettualmente durante le lezioni):

- ***Algorithm selection***: Un problema di *bandit* con *expert advice*, usando un *solver* esistente; oppure un problema di *bandit*, usando informazioni parziali e *bounds* sconosciuti di perdite.
- ***Hyperparameter optimization***: Ottimizzazione di iperparametri come un *Multi-armed Bandit* (infinitamente *armed*) non stocastico di pura esplorazione in cui risorse predefinite, come iterazioni, campioni di dati o funzionalità sono allocate a configurazioni campionate casualmente.

[46] Djallel Bouneffouf e Irina Rish. «A Survey on Practical Applications of Multi-Armed and Contextual Bandits». Inglese. In: (apr. 2019). Documento locale in pdf. eprint: 1904.10040. URL: https://arxiv.org/abs/1904.10040.

- ***Feature selection***: Fare previsioni accurate usando solo un piccolo numero di *feature* attive usando l'algoritmo epsilon-greedy.
- ***Bandits* per *active learning***: Selezione di *samples* non *labeled* più utili per formare un modello predittivo.
- ***Clustering***: *Clustering* collaborativo, paradigma dell'apprendimento automatico.
- ***Reinforcement learning***: Massimizzazione *rewards* degli *agents*.

Capitolo 7

Considerazioni personali

Icona del Capitolo 7[2]

7.1 Note personali

La preparazione di questo elaborato ha contribuito ad un generale aumento della conoscenza personale sul tema oggetto di studio. L'intero iter di lettura, ricerca, studio, sperimentazione e *reviewing* non solo ha permesso di padronneggiare le nozioni base sull'argomento, ma ha permesso di capire decisamente meglio anche altri argomenti in ambito *machine learning*.

La soddisfazione per questa opportunità offerta e il lavoro fatto, è tanta.

7.2 *Tools* usati

Alcuni *tools* usati durante la preparazione di questo elaborato:

- **Calibre**: Gestione e lettura ebook e *papers*.
- **`gedit`**: *Text editing*, veloce lettura e scrittura LaTeX e Python, Matlab; Veloce visualizzazione di *dataset* di grosse dimensioni.
- **`git`**: Controllo di versione.
- **GitHub**: *Online repository* del codice sorgente degli autori.
- **Inkscape**: Modifica immagini vettoriali e documenti in formato `pdf`.
- **`LyX`** e **Kile**: Scrittura documento in LaTeX in sistema `GNU/Linux`.
- **Matlab**: Esecuzione del codice Matlab.
- **Overleaf**: Scrittura in Latex dell'elaborato, gestione dei riferimenti bibliografici.
- **PyCharm**: Sviluppo codice, IDE Python.

7.3 Librerie usate

Alcune librerie di Python usate durante la preparazione di questo elaborato:

- **`Math`**: Calcoli generici matematici.
- **`NumPy`**: Calcoli con vettori e matrici.
- **`matplotlib`**: *Plotting* di grafici.
- **`Pandas`**: Manipolazione e *querying* di dati.

Glossario

adattabile in grado di adattarsi 8, 9, 17, 19, 32, 77

adattivo in grado di adattarsi 5, 11–13, 15, 16, 18, 19, 23, 32, 77

arithmetic grid griglia aritmetica 35, 53, 74

arm braccio, scelta, levetta di una macchina v, xiii, 5–7, 12, 13, 17–19, 23, 26–32, 35, 42, 43, 53, 54, 56, 57, 74–78

articolo articolo di ricerca scientifica 3

bandit bandito 13, 15–18, 22, 23, 56, 57, 76–79

batch gruppo, blocco v, xiii, 5–9, 12, 14, 15, 19, 22–24, 26–29, 32, 33, 35, 42, 43, 53–55, 75, 77

batch learning apprendimento a blocchi, gruppi di informazioni 5, 6

batched suddiviso in gruppi xiii, 5, 7, 12, 13, 23

bound limite 8, 28, 78

cleanup pulizia 55

crowd massa di persone 6, 20

crowdsourcing attività di raccolta di idee, suggerimenti, opinioni o di soldi da un gruppo, una massa di persone 6, 20, 21

dataset una collezione di dati 4, 55–57, 74, 77, 81

divergenze Kullback-Leibler misura di come una distribuzione di probabilità è diversa da un'altra distribuzione di probabilità di riferimento 25

economia economia 11

exploitation fare uso e beneficiare delle informazioni già in possesso 14, 78

exploration esplorare per raccogliere nuove informazioni 14, 78

gap divario xiii, 6, 7, 18, 30, 32

geometric grid griglia geometrica 35, 53, 74

grid rete, griglia 6, 12, 25–30, 32, 35, 42

grid adattabile griglia adattabile ix, 6, 8, 9, 25, 28, 30, 32

grid statica griglia statica ix, 6, 8, 9, 25, 26, 30, 32

informatica informatica 11

ingegneria ingegneria 11

learner persona, agente, macchina che apprende 5

learning l'attività di apprendimento 6

limite inferiore limite inferiore 32, 35

limite superiore limite superiore 35

lower bound limite inferiore ix, 5, 7–9, 25, 30, 31

machine learning branca dell'intelligenza artificiale che raccoglie un insieme di metodi per l'apprendimento automatico 4, 5, 78

marketing ambito delle azioni destinate alla vendita di prodotti o servizi in cui si decide se esplorare o *exploitare* le conoscenze sui segmenti di mercato 6

minimax metodo nella teoria delle decisioni per minimizzare la massima perdita possibile ix, 8, 16, 21, 23, 26, 27, 29, 30

minimax grid griglia minimax 35, 42, 74

minimax regret strategia di minimizzazione del maximum regret xiii, 5, 7–9, 12, 14, 25, 27, 30, 32, 77

online in linea, su internet 4

online learning Apprendimento online, ricavando informazioni da fonti su internet 5, 7, 77

paper articolo di ricerca scientifica 3–6, 14, 25, 29, 44, 53, 77, 81

performance prestazione v, 6, 35, 43, 54, 75

policy politica v, ix, xiii, 4, 5, 7–9, 14, 16–18, 23, 24, 26, 28–33, 35, 43, 54, 55, 57, 75, 78

proceeding atto di una conferenza 3

query interrogare, interrogazione 6

random casuale, casualmente v, 4, 16, 18, 20, 32, 35, 44, 54, 55

rate-optimal tasso di ottimalità, rate o tasso ottimale 5

regret differenza tra il reward ottimale e quello reale v, ix, xiii, xiv, 5, 7–9, 11, 13, 14, 16–18, 22–24, 26, 28–33, 35, 42, 43, 53, 54, 57, 74–77

regret dipendente dal problema differenza tra il reward ottimale e quello reale dipendente dal problema xiii, 7–9, 25, 27, 28, 30, 32

reinforcement learning apprendimento automatico mediante la scelta delle azioni in modo da raggiungere obiettivi interagendo con l'ambiente 22

reviewer ricercatore scientifico che valutano articoli scientifici di un Journal 55

reward ricompensa xiii, 6, 14, 15, 17, 18, 23, 24, 26, 31, 33, 35, 56, 57, 76, 78, 79

ricerca operativa ricerca operativa 11

round turno 6

set insieme 6

simulation selection problem Problema di decidere quanto a lungo simulare prima di scegliere o rifiutare un'alternativa 6

statistica statistica 11

stochastic bandit giocatore, bandito stocastico 6, 11, 14, 17, 18

switching cost costo di cambiare, o di cambiare scelta/e 12

test clinico test diagnostico per il controllo dello stato di salute 6

tool mezzo, strumento 4

transpiling traduzione da un linguaggio di programmazione a un altro vii, 44, 49–53, 57

upper bound limite superiore 7, 8, 29, 30

Riferimenti bibliografici

[1] Università degli Studi di Bergamo. *Logo UniBG*. Online. Mag. 2020. URL: https://unibg.it/ (Citato a pagina 3).

[2] UXWing. *Icons*. Online. Mag. 2020. URL: https://uxwing.com/ (Citato alle pagine 3, 5, 11, 25, 35, 77, 81).

[3] Zijun Gao, Yanjun Han, Zhimei Ren e Zhengqing Zhou. «Batched Multi-armed Bandits Problem». Inglese. In: *Advances in Neural Information Processing Systems 32*. A cura di H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox e R. Garnett. Documento locale in pdf. Curran Associates, Inc., 2019, pagine 503–513. URL: http://papers.nips.cc/paper/8341-batched-multi-armed-bandits-problem.pdf (Citato alle pagine 3, 5).

[4] Vianney Perchet, Philippe Rigollet, Sylvain Chassang e Erik Snowberg. «Batched bandit problems». Inglese. In: *Annals of Statistics* 44.2 (apr. 2016). Documento locale in pdf, pagine 660–681. DOI: 10.1214/15-AOS1381. URL: https://projecteuclid.org/euclid.aos/1458245731 (Citato alle pagine 5, 12, 14, 27, 42, 53).

[5] William R. Thompson. «On the likelihood that one unknown probability exceeds another in view of the evidence of two samples». Inglese. In: *Biometrika* 25.3-4 (dic. 1933). Documento locale in pdf, pagine 285–294. ISSN: 0006-3444. DOI: 10.1093/biomet/25.3-4.285. eprint: https://academic.oup.com/biomet/article-pdf/25/3-4/285/513725/25-3-4-285.pdf (Citato alle pagine 6, 14).

[6] Herbert Robbins. «Some aspects of the sequential design of experiments». Inglese. In: *Bulletin of the American Mathematical Society* 58.5 (set. 1952), pagine 527–535. URL: https://projecteuclid.org:443/euclid.bams/1183517370 (Citato alle pagine 6, 15).

[7] Aniket Kittur, Ed H. Chi e Bongwon Suh. «Crowdsourcing User Studies with Mechanical Turk». Inglese. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Documento locale in pdf. Florence, Italy: Association for Computing Machinery, apr. 2008, pagine 453–456. ISBN: 9781605580111. DOI: 10.1145/1357054.1357127 (Citato alle pagine 6, 15).

[8] Dimitris Bertsimas e Adam J. Mersereau. «A Learning Approach for Interactive Marketing to a Customer Segment». Inglese. In: *Operations Research* 55.6 (nov. 2007). Documento locale in pdf, pagine 1120–1135. ISSN: 0030-364X. DOI: 10.1287/opre.1070.0427. URL: https://www.researchgate.net/publication/220244453 (Citato alle pagine 6, 15).

[9] Stephen E. Chick e Noah Gans. «Economic Analysis of Simulation Selection Problems». Inglese. In: *Management Science* 55.3 (mar. 2009). Documento locale in pdf, pagine 421–437. DOI: 10.1287/mnsc.1080.0949. URL: https://www.researchgate.net/publication/227447666 (Citato alle pagine 6, 15, 16).

[10] Walter Vogel. «A Sequential Design for the Two Armed Bandit». Inglese. In: *Annals of Mathematical Statistics* 31.2 (giu. 1960). Documento locale in pdf, pagine 430–443. DOI: 10.1214/aoms/1177705906 (Citato alle pagine 7, 11, 16).

[11] Tze Leung Lai e Herbert Robbins. «Asymptotically efficient adaptive allocation rules». Inglese. In: *Advances in applied mathematics* 6.1 (mar. 1985). Documento locale in pdf, pagine 4–22. ISSN: 0196-8858. DOI: 10.1016/0196-8858(85)90002-8 (Citato alle pagine 7, 8, 11, 16).

[12] Jean-Yves Audibert e Sébastien Bubeck. «Minimax policies for adversarial and stochastic bandits». Inglese. In: *COLT*. Documento locale in pdf. Montreal, Canada, giu. 2009, pagine 217–226. DOI: 10.1016/0196-8858(85)90002-8. URL: https://hal-enpc.archives-ouvertes.fr/hal-00834882 (Citato alle pagine 7, 8, 11, 16).

[13] Sébastien Bubeck, Vianney Perchet e Philippe Rigollet. «Bounded regret in stochastic multi-armed bandits». Inglese. In: *Proceedings of the 26th Annual Conference on Learning Theory*. A cura di Shai Shalev-Shwartz e Ingo Steinwart. Volume 30. Proceedings of Machine Learning Research. Documento locale in pdf. Princeton, NJ, USA, giu. 2013, pagine 122–

134. eprint: 1302.1611. URL: http://proceedings.mlr.press/v30/Bubeck13.html (Citato alle pagine 7, 11, 18).

[14] Sébastien Bubeck e Nicolò Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Inglese. Documento locale in pdf. 2012. DOI: 10.1561/2200000024 (Citato alle pagine 11, 16).

[15] Apostolos N. Burnetas e Michael N. Katehakis. «Optimal Adaptive Policies for Markov Decision Processes». Inglese. In: *Mathematics of Operations Research* 22.1 (feb. 1997). Documento locale in pdf, pagine 222–255. ISSN: 0364765X, 15265471. DOI: 10.1287/moor.22.1.222. URL: http://www.jstor.org/stable/3690147 (Citato alle pagine 11, 16, 17).

[16] Peter Auer, Nicolò Cesa-Bianchi e Paul Fischer. «Finite-time Analysis of the Multiarmed Bandit Problem». Inglese. In: *Machine Learning* 47.2 (mag. 2002). Documento locale in pdf, pagine 235–256. ISSN: 1573-0565. DOI: 10.1023/A:1013689704352 (Citato alle pagine 11, 24, 42).

[17] Jean-Yves Audibert, Rémi Munos e Csaba Szepesvári. «Exploration–exploitation tradeoff using variance estimates in multi-armed bandits». Inglese. In: *Theoretical Computer Science* 410.19 (apr. 2009). Documento locale in pdf, pagine 1876–1902. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.01.016. URL: http://www.sciencedirect.com/science/article/pii/S030439750900067X (Citato alle pagine 11, 17).

[18] Jean-Yves Audibert e Sébastien Bubeck. «Regret Bounds and Minimax Policies under Partial Monitoring». Inglese. In: *Journal of Machine Learning Research* 11.94 (dic. 2010). Documento locale in pdf, pagine 2785–2836. ISSN: 1532-4435. URL: http://jmlr.org/papers/v11/audibert10a.html (Citato alle pagine 12, 17).

[19] Peter Auer e Ronald Ortner. «UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem». Inglese. In: *Periodica Mathematica Hungarica* 61.1 (set. 2010). Documento locale in pdf, pagine 55–65. ISSN: 1588-2829. DOI: 10.1007/s10998-010-3055-6 (Citato alle pagine 12, 17, 18).

[20] Aurélien Garivier e Olivier Cappé. «The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond». Inglese. In: *Proceedings of the 24th Annual Conference on Learning*

Theory. A cura di Sham M. Kakade e Ulrike von Luxburg. Volume 19. Proceedings of Machine Learning Research. Documento locale in pdf. Budapest, Hungary, giu. 2011, pagine 359–376. URL: http://proceedings.mlr.press/v19/garivier11a.html (Citato alle pagine 12, 18).

[21] Vianney Perchet e Philippe Rigollet. «The multi-armed bandit problem with covariates». Inglese. In: *Annals of Statistics* 41.2 (apr. 2013). Documento locale in pdf, pagine 693–721. ISSN: 0090-5364. DOI: 10.1214/13-aos1101 (Citato alle pagine 12, 23, 26).

[22] Nicolò Cesa-Bianchi, Ofer Dekel e Ohad Shamir. «Online Learning with Switching Costs and Other Adaptive Adversaries». Inglese. In: *Advances in Neural Information Processing Systems 26*. A cura di C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani e K. Q. Weinberger. NIPS'13. Documento locale in pdf. Curran Associates, Inc., 2013, pagine 1160–1168. eprint: 1302.4387. URL: https://papers.nips.cc/paper/5151-online-learning-with-switching-costs-and-other-adaptive-adversaries (Citato alle pagine 12, 18).

[23] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak e Xiaojin Zhu. «Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls». Inglese. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. A cura di Arthur Gretton e Christian C. Robert. Volume 51. Proceedings of Machine Learning Research. Documento locale in pdf. Cadiz, Spain, mag. 2016, pagine 139–148. URL: http://proceedings.mlr.press/v51/jun16.html (Citato alle pagine 12, 19).

[24] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi e Sanjeev Khanna. «Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons». Inglese. In: *Proceedings of the 2017 Conference on Learning Theory*. A cura di Satyen Kale e Ohad Shamir. Volume 65. Proceedings of Machine Learning Research. Documento locale in pdf. Amsterdam, Netherlands, lug. 2017, pagine 39–75. URL: http://proceedings.mlr.press/v65/agarwal17c.html (Citato alle pagine 12, 13, 14, 19).

[25] John Duchi, Feng Ruan e Chulhee Yun. «Minimax Bounds on Stochastic Batched Convex Optimization». Inglese. In: *Proceedings of the 31st Conference On Learning Theory*. A cura di Sébastien Bubeck, Vianney Perchet e Philippe Rigollet. Volume 75. Proceedings of Machine Learning Research. Documento locale in pdf. Lug. 2018, pagine 3065–3162. URL:

http://proceedings.mlr.press/v75/duchi18a.html (Citato alle pagine 13, 14, 22).

[26] Leslie G. Valiant. «Parallelism in Comparison Problems». Inglese. In: *SIAM Journal on Computing* 4.3 (1975). Documento locale in pdf, pagine 348–355. DOI: 10.1137/0204030 (Citato alle pagine 13, 19).

[27] Béla Bollobás e Andrew Thomason. «Parallel sorting». Inglese. In: *Discrete Applied Mathematics* 6.1 (1983). Documento locale in pdf, pagine 1–11. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(83)90095-1 (Citato alle pagine 13, 19, 20).

[28] Noga Alon e Yossi Azar. «Sorting, approximate sorting, and searching in rounds». Inglese. In: *SIAM Journal on Discrete Mathematics* 1.3 (ago. 1988). Documento locale in pdf, pagine 269–280. ISSN: 0895-4801. DOI: 10.1137/0401028 (Citato alle pagine 13, 20).

[29] Uriel Feige, Prabhakar Raghavan e Eli Upfal. «Computing with Noisy Information». Inglese. In: *SIAM Journal on Computing* 23 (ott. 1994). Documento locale in pdf, pagine 1001–1018. ISSN: 0097-5397. DOI: 10.1137/S0097539791195877 (Citato alle pagine 13, 20).

[30] Susan Davidson, Sanjeev Khanna, Tova Milo e Sudeepa Roy. «Top-k and Clustering with Noisy Comparisons». Inglese. In: *ACM Transactions on Database Systems* 39.4 (dic. 2014). Documento locale in pdf, pagine 1–39. ISSN: 0362-5915. DOI: 10.1145/2684066 (Citato alle pagine 13, 20, 21).

[31] Mark Braverman, Jieming Mao e S. Matthew Weinberg. «Parallel Algorithms for Select and Partition with Noisy Comparisons». Inglese. In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Documento locale in pdf. Association for Computing Machinery, 2016, pagine 851–862. ISBN: 9781450341325. DOI: 10.1145/2897518.2897642 (Citato alle pagine 13, 21).

[32] Arkadii Semenovich Nemirovsky e David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. Inglese. Wiley-Interscience series in discrete mathematics. John Wiley & Sons, 1983. 388 pagine. ISBN: 9780471103455. URL: http://www.worldcat.org/oclc/252233116 (Citato alle pagine 13, 21).

[33] Arkadij Semenovič Nemirovskij e David Borisovič Judin. *Ložnost' zadač i èffektivnost' metodov optimizacii*. Russo. Nauka, 1979. 384 pagine (Citato alle pagine 14, 21).

[34] Alekh Agarwal, Martin J. Wainwright, Peter L. Bartlett e Pradeep K. Ravikumar. «Information-theoretic lower bounds on the oracle complexity of convex optimization». Inglese. In: *Advances in Neural Information Processing Systems 22*. A cura di Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams e A. Culotta. Documento locale in pdf. Curran Associates, Inc., 2009, pagine 1–9. eprint: 1009.0571. URL: https://papers.nips.cc/paper/3689-information-theoretic-lower-bounds-on-the-oracle-complexity-of-convex-optimization (Citato alle pagine 14, 21, 22).

[35] Ohad Shamir. «On the Complexity of Bandit and Derivative-Free Stochastic Convex Optimization». Inglese. In: *Proceedings of the 26th Annual Conference on Learning Theory*. A cura di Shai Shalev-Shwartz e Ingo Steinwart. Volume 30. Proceedings of Machine Learning Research. Documento locale in pdf. Princeton, NJ, USA, giu. 2013, pagine 3–24. URL: http://proceedings.mlr.press/v30/Shamir13.html (Citato alle pagine 14, 22).

[36] Eyal Even-Dar, Shie Mannor e Yishay Mansour. «Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems». Inglese. In: *Journal of Machine Learning Research* 7 (dic. 2006). Documento locale in pdf, pagine 1079–1105. ISSN: 1532-4435 (Citato alle pagine 22, 23, 26).

[37] Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian e Vahab S. Mirrokni. «Regret Bounds for Batched Bandits». Inglese. In: *arXiv: Data Structures and Algorithms* (2019). Documento locale in pdf. URL: http://arxiv.org/abs/1910.04959v2 (Citato alle pagine 23, 24, 28).

[38] T. M. Cover e J. A. Thomas. *Elements of Information Theory*. Inglese. Wiley Series in Telecommunications and Signal Processing. Documento locale in pdf. Wiley, 2006. 776 pagine. ISBN: 9780471748816. URL: https://books.google.it/books?id=0QuawYmc2pIC (Citato alle pagine 24, 31).

[39] Zijun Gao, Yanjun Han, Zhimei Ren e Zhengqing Zhou. *Batched Multi-armed Bandits Problem Source code*. Inglese. Source code of the experiments. 2020. URL: https://github.com/Mathegineer/batched-bandit/ (Citato a pagina 36).

[40] Alphabet Inc. *Kaggle website*. Inglese. Kaggle, una susidiary di Google LLC, è una community online di data scientist e professionisti del machine learning. Kaggle consente agli utenti di trovare e pubblicare datasets, esplorare e costruire modelli in un ambiente di data science

basato sul web, lavorare con altri data scientist e ingegneri del machine learning e partecipare a concorsi per risolvere data science challenges. 2020. URL: https://www.kaggle.com/ (Citato a pagina 55).

[41] GroupLens Research. *GroupLens website*. Inglese. GroupLens Research è un laboratorio di ricerca sull'interazione human–computer nel Dipartimento di Informatica e Ingegneria dell'Università del Minnesota. 2020. URL: http://www.grouplens.org/ (Citato a pagina 55).

[42] GroupLens Research. *MovieLens website data*. Inglese. GroupLens Research ha raccolto e reso disponibili datasets del sito web MovieLens. 2020. URL: https://grouplens.org/datasets/movielens/ (Citato a pagina 55).

[43] MovieLens. *MovieLens 1M movie ratings*. Inglese. Stable benchmark dataset. 1 million ratings from 6000 users on 4000 movies. Feb. 2003. URL: https://grouplens.org/datasets/movielens/1m/ (Citato alle pagine 55, 56).

[44] F. Maxwell Harper e Joseph A. Konstan. «The MovieLens Datasets: History and Context». Inglese. In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (dic. 2015). Documento locale in pdf, pagine 235–256. ISSN: 2160-6455. DOI: 10.1145/2827872 (Citato alle pagine 55, 56).

[45] MovieLens. *MovieLens 10M movie ratings*. Inglese. Stable benchmark dataset. 10 million ratings and 100000 tag applications applied to 10000 movies by 72000 users. Gen. 2009. URL: https://grouplens.org/datasets/movielens/10m/ (Citato a pagina 56).

[46] Djallel Bouneffouf e Irina Rish. «A Survey on Practical Applications of Multi-Armed and Contextual Bandits». Inglese. In: (apr. 2019). Documento locale in pdf. eprint: 1904.10040. URL: https://arxiv.org/abs/1904.10040 (Citato a pagina 78).