ENGINEERING PHYSICS | FINAL THESIS

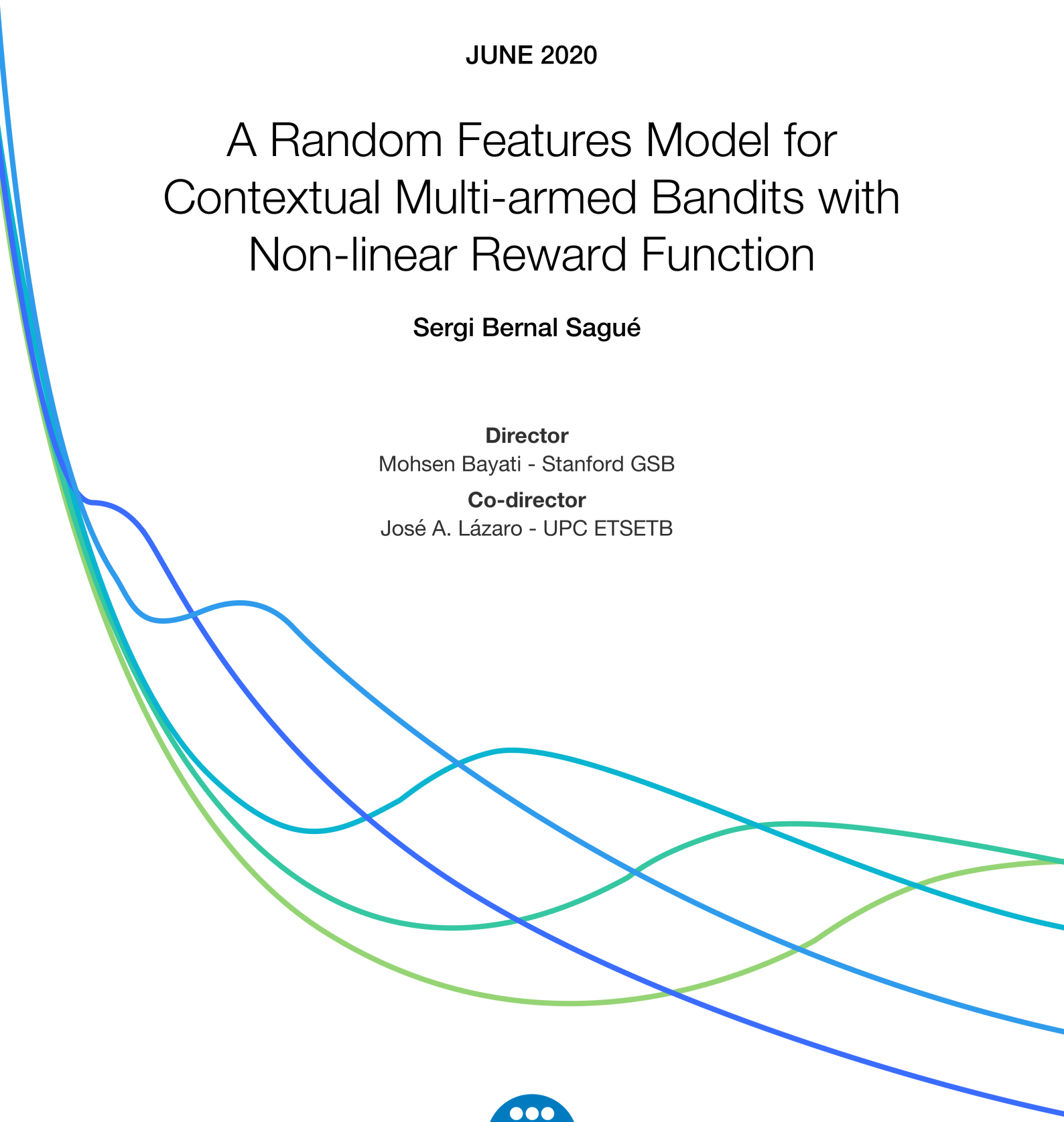
JUNE 2020

A Random Features Model for Contextual Multi-armed Bandits with Non-linear Reward Function

Sergi Bernal Sagué

Director
Mohsen Bayati - Stanford GSB

Co-director
José A. Lázaro - UPC ETSETB



Preamble

I have been interested in improving decision making in healthcare since I co-founded Elenytics in 2016. Back then, we used indoor location data to improve efficiency in hospitals. My goal in this project, is to set a general method to improve decision making using patient data and Contextual Bandits.

Aknowledgments

This project wouldn't have been possible without the help and guidance from professor Mohsen Bayati. Although the first months of my project consisted only in learning about the field, he kept supporting me and dedicating time to guide my learning process. Also, most of the modifications that led to the final algorithm in this project where proposed by professor M. Bayati.

I also want to thank my co-workers at Polyup, without whom I wouldn't have had the opportunity of doing any of this. Specially to professor Yahya Tabesh, who introduced me to professor Bayati and Zohre Elahian, who hosted me during my presential research at Stanford. I also want to thank professor José A. Lázaro, for supporting me in every step of the way and giving me trust and freedom to work on the project.

I can't finish without thanking Julia, Mr. Robbins, my parents and friends, without their constant support it wouldn't have been possible. This project is dedicated to all of you.

Contents

1 Introduction

Dynamic decisions under uncertainty are present across industries, from tailoring content to customer's preferences to analysing clinical trials on new treatments. One of the most popular setups to approach these problems are the Bandit Algorithms. They were introduced by Thompson (1933) and are becoming a main field of research across industries. In the bandit framework, a decision-maker is presented with a set of options and it should choose the one that maximizes its reward, i.e. the best available option. The decision-maker is given such reward after every iteration, allowing a learning process that will adjust its criteria for the next decision to take. We present two different scenarios, depending on the nature of each iteration. In the first case, every iteration has the same environment (conditions) as the previous one, meaning that the only information that affects the decision-makers choice is the outcomes (rewards) of the previous choices. The most famous example of this setup would be a user in a casino playing the different slots machines to try to learn the reward distribution of each of them. In the second case, every iteration is associated with a unique vector that conditions the outcome of the choice made by the decision maker. An example of an application would be the web recommendation systems tailored to specific user preferences. In section 1.1 we go through the main current application of such algorithms.

Mathematical formulation. In this setup, an agent has the possibility to choose between K actions (arms), each with it's associated d-dimensional vector of unknown parameters β_i. Each arm has an uncertain reward distribution that the agent aims to discover through the d-dimensional unknown vector for each arm. At every round, the agent is presented with a context $\mathbf{X}_t$ and chooses an action (arm) that generates a stochastic reward conditioned to the action take but also to the feature vector in that specific context. The goal is to maximize the cumulative rewards over T rounds.

As opposed to the traditional multi-armed bandit, where the learner has no access to arm features, and therefore, only competes with choosing the best arm in the hindsight, contextual bandits use each specific context action features to improve the average payoff over time.

The most studied model in this topic are linear contextual bandits, assuming that the distribution of the reward is linear in the feature vector. One of the most common application of this setup includes customizing web content for each user using their search history and the success of previous trials with users that have similar preferences than the agent (Li et al. (2010)). In this particular linear model, the agent will have K actions (arms) with uncertainty rewards. Each arm associated with the d-dimensional vector of unknown parameters $\beta_i \in \mathbb{R}^d$. At each time t, the agent will see an individual with a context vector $\mathbf{X}_t^\top \in \mathbb{R}^d$. After choosing an arm i, the agent will observe a linear reward of

$$\mathbf{Y}_t = \mathbf{X}_t^\top \beta_i + \epsilon_{i,t} \tag{1.1}$$

with $\epsilon_{i,t}$ as an independent sub-Gaussian random variable.

This context vector can contain different types of information depending on the application of the algorithm. For example, in the previous setting, where the learner seeks to tailor web content to user's preferences, the context vector $\mathbf{X}_t$ would contain information on their previous searches, clicks on the website, content preferences and any other feature specific to the user that could help identify which is the best option to display (in our setting, which arm to choose). Another application would be choosing the best treatment for a patient after looking at its specific context vector containing previous pathologies, physical characteristics, sympthoms and any other relevant information that could affect the decision on the type of treatment.

Although the linear approach has demonstrated to be successful in multiple setups, both in synthetic data (Auer (2002)) and real life applications (Lei et al. (2017)), it often fails in practice due to the non-linear nature of the true reward function. A reason that motivated the study of nonlinear or non-parametric contextual bandits:

$$\mathbf{Y}_t = \mu(\mathbf{X}_t^\top \beta_i) + \epsilon_{i,t} \tag{1.2}$$

with μ as an inverse link function (Filippi et al. (2010), Li et al. (2017)).

In any setting, the decision-maker's goal is to learn the arm parameters (β_i) to maximize the cumulative reward over a period of time T. There are many policies that we will introduce later in the project that achieve to be asymptotically optimal in this kind of settings.

Most of these algorithms are based on the idea of the Upper Confidence Bound or UCB (T. Lai & Robbins (1985), T. L. Lai (1987), Katehakis & Robbins (1995)), but there are other lines of research using Thompson Sampling (Thompson (1933), Agrawal & Goyal (2012), Russo & Roy (2013)) and ϵ-greedy methods (Goldenshluger & Zeevi (2013), Bastani et al. (2019a)). The computational cost and performance of each varies depending on the nature of the observed context, thus making them optimal only in specific settings.

The main challenge that all these algorithms face is the problem of the exploration-exploitation trade-off. While the decision-maker would like to explore all the available options, i.e. K possible arms, it also wants to exploit each of them to learn the reward distribution. Finding the optimal balance to explore all the options while exploiting the ones that seem to have a higher reward is the secret sauce that can make an algorithm outperform the rest. Algorithms that explore too much usually have a poor performance due to not learning well the arm parameters.

In the Ordinary Least Squares (OLS) bandit (ϵ-greedy) from Sutton & Barto (1998) the proportion of time allocated for exploration and exploitation is fixed and defined prior to running the experiment. In our case, we explore algorithms that, assuming randomness in the observed context, create a natural exploration without the need to define a fixed proportion of time for such purpose.

In this research, we focus on applying methods that are proven to work on similar settings. From Ordinary Least Squares approximation to Machine Learning techniques, in order to improve the efficiency of our Exploration Free Greedy Bandit algorithms. We want to define the conditions that the observed context has to comply in order to achieve a state of the art accuracy with our proposed algorithm. That being said, our ultimate goal is to proof it's

performance in real and meaningful datasets.

During the execution of this project the pandemic of the COVID-19 has impacted the entire planet and forced all the data science community to work together towards solutions that could allow world leaders and doctors to adopt better solutions on how to react to it. One of the main problems has been the allocation of tests. The scarcity of testing devices has been one of the main issues that governments have faced. In fact, South Korea was the only country that managed to have an effective testing system and isolate the infected patients on time before they were able to spread the virus. The problem of deciding weather to test or not test a person depending on their risk factor, symptoms, travel history, physical characteristics or any other relevant parameter would be a perfect example of a 2 arm contextual bandit problem. It is our goal to ultimately apply the algorithm to a similar database that contains such kind of information to see if the algorithm would have been an optimal tool to allocate testing when the availability was limited.

1.1 Related literature

There are many articles on contextual bandits nowadays, but we would like to first refer the reader to Lattimore & Szepesvári (2020) for a basic and general understanding on how does the contextual bandit setup works. It also uses the same regret analysis as we do in the project, explaining how will the success of the algorithms be measured.

The linear contextual bandit problem was first introduced by Auer (2000) using the LinRel (Linear Associative Reinforcement Learning) algorithm, which uses single value decomposition to obtain an estimate of confidence. It was later improved by the OFUL algorithm from Dani et al. (2008) and the LinUCB (Upper Confidence Bound) from Chu et al. (2011). There are also some algorithms that focus on contextual bandit problems with non linear reward functions, like the KernelUCB from Valko et al. (2013), a kernelized non-linear version of the linear Upper Confidence Bound algorithm, or the NeuralBandit algorithm from Zhou et al. (2019) where multiple neural networks are trained to predict the value of the rewards after knowing the specific context. Another non-linear approach that is of our interest is the Oracle-based Algorithm, that turns the contextual bandit problem into several supervised learning problems (Agarwal et al. (2014)). We will be using the Oracle-based algorithm to compare it with the performance of the one presented in this project.

The related literature for each of the main applications of such algorithms could be organized in the following groups:

1. *Resource Allocation.* It is probably the most promising application for bandits, choosing where to allocate resources when they are scarce (Dagan & Crammer (2018)).It is also the application in which we propose the use of contextual bandits in relation to COVID-19 crisis, specifically in terms of testing allocation. We will further discuss it in 5.

2. *A/B Testing and Ads Placement.* This is the most widely used application for bandit algorithms. Specially for contextual bandits, as the actions taken by the algorithm depend on the specific user profile. It can be applied in the online service economy

(Chu et al. (2011) and Langford & Zhang (2007)) or ads placement (Schwartz et al. (2017)). In the previous settings, the algorithm chooses to display a certain option to the user and gets a reward when the user clicks on it. In the contextual bandit framework, the action of choosing the a specific Ad (arm) depends on the user profile.

3. *Dynamic Pricing.* In this case we have the situation where a company wants to automatically optimize pricing to match each users valuation. At every iteration of the bandit algorithm a new user comes, the decision-maker decides which price to show and then observes if the user buys or not. What makes this setup complicated and interesting at the same time is that the decision-maker only observes if the price offered is less than the valuation of the user, instead of the actual valuation of the user. That is, if there is a product bought for 100 EUR by the user, we only know that his valuation was higher than 100 EUR, but we will never know if the user would have bought the product anyways at a 150 EUR or 200 EUR price (Cohen et al. (2016), Qiang & Bayati (2016), Ban & Keskin (2017), Bastani et al. (2019b)).

4. *Clinical Trials.* Although we are not aware that there has been any implementation to date in which the bandit algorithms have actually been used in clinical trials, it is the option that is gaining more popularity among the regulatory administrations to accelerate the launch of new drugs to the market (Villar et al. (2015)).

Greedy Algorithms. One of the most studied algorithms in this field is the Epsilon Greedy (ϵ-greedy). In this case, after an initial period of exploration, the algorithm exploits the best option for a fixed percentage of trials (1-ϵ) and dedicates the rest of the time to exploitation. This setup is explained in Goldenshluger & Zeevi (2013). What motivated our study is the poor performance of this setup due to the fixed assignment of exploitation and exploration proportion. Specially in contextual bandits, it doesn't make sense to fix the exploration parameter because each context (iteration) is different. For that reason, we focus on exploration-free greedy bandit algorithms, in which the exploration becomes natural instead of fixed. The main article that explores this specific kind of algorithms was published by Bastani et al. (2017). It focuses on the Exploration Free Greedy Algorithms and compares the algorithm with the ones previously mentioned. In fact, one of our goals will be to demonstrate that our proposed algorithm can beat the Greedy First algorithm proposed in the paper.

Random Features Another line of work that we want to refer to the reader is about the double descent curve phenomena in Machine Learning setups. Specifically the work by Mei & Montanari (2019) where they describe the method we have used to generate random features and improve the performance of the algorithm in the over-parameterized regime. The paper itself is extremely dense in the theory behind the phenomena, trying to give an explanation to the fact that adding random features (pure noise) to a multiple regression can actually improve its performance.

1.2 Main contributions and organization of this project

In this project we begin by introducing the bandit framework, starting with the basic stochastic bandits and it's main algorithms. We will then introduce the contextual bandit framework, from which we will derive our proposed algorithm. We will focus on the Greedy Bandit proposed by Bastani et al. (2017) as the base from which we can build on.

We will also introduce the Machine Learning techniques from which we have derived the modifications of our algorithm, focusing on the effects of Random Features presented by Mei & Montanari (2019). Then we will present the final version of the algorithm and theoretically evaluate the advantages brought by each modification. Finally, we will evaluate the results in both synthetic and real data, specially comparing in the same framework presented in Bastani et al. (2017).

From the later results we will see that our algorithm provides an outstanding improvement of accuracy in setups where the number of iterations is much larger than the number of features with a 2-armed framework. The random features transformation seems to provide more room for improvement when the other algorithms achieve a steady cumulative regret bound. We also discover that sequentially adding random features can improve the behavior of the algorithm when the number of features is large.

We also find an improvement of performance in the over-parameterized regime, where the number of features of each arm is larget than the number of iterations (observations). This is provided by the random features of the modified context and represents the main innovation of this project in the contextual bandit framework.

We also realize that the value of λ when applying ridge regression to approximate our weights $\tilde{\beta}_{i,t}$ is specific to each context $\mathbf{X}_t$ in every iteration. Ideally, our algorithm would have to find the optimal λ value through Cross Validation at each time t. However, we find it computationally impossible when as the number of iterations grows.

In summary, we present a Bandit algorithm that achieves a state of the art accuracy in setups with a strongly non-linear true reward function. It's main contribution relies on the use of random features to increase the accuracy with high-dimensional feature vectors and a limited number of instances.

2 Fundamentals of Bandit Algorithms

Although the first application for Bandits was supposed to be in clinical trials (Thompson (1933)), the name "bandit" comes from the experiment ran by Bush & Mosteller (1953) in which humans were presented with a "two-armed bandit machine" and they had to choose either to pull the right arm or the left one. Each of the machines was offering a random pay-off from a distribution that was completely unknown for the human. From that lever-operated slot machine is where the name "bandit" comes from. Generalizing that specific case, we can state that Bandit Algorithms offer a simple model to face Decision-making under uncertainty. In the following chapters we will explore the fundamentals of two large groups of Bandit Algorithms:

- **Stochastic Bandits**. Where the decision and output of each iteration only depends on the distribution pay-offs of each option.
  - *Contextual Bandits*. Where each decision and output depends on the profile of the user.
- **Adversarial Bandits**. In this case the decision-maker does not make any probabilistic assumptions regarding the distribution of the reward. Instead, those rewards are generated by an adversary, from which we can't make almost any assumption on the rewards distribution.
- **Markovian Bandits**. In this case the reward distribution of the arm is sampled from a Markov Chain[1] from some underlying state space. The process to analyse these kind of MAB are quite different from the tools used in Stochastic and Adversarial MABs. Significant work on this direction includes Burnetas & Katehakis (1997) and Ortner (2010).

2.1 The stochastic bandit framework

A Bandit Problem consists on a sequential game between the decision-maker and a certain environment. At every time $t \in [T]$, with T being the total number of rounds, the decision-maker chooses an action $i_t \in K$ with K as the total number of options that the decision-maker has. After choosing, the environment shows the reward $\mathbf{Y}_t$ and makes a guess on the mean reward $\tilde{\mu_i}$ of such arm. As previously stated, the different set of actions i_t is usually referred to as Arms. Then a "K-armed Bandit" would be a Bandit Problem with K different actions. As we previously stated, the bandit problem is a sequential game, that means, the decision

[1] A Markov Chain is a sequence of random elements with the property that given the last element of the series, the history is completely irrelevant to predict the next element.

maker can only make the decision taking into account the history $H_{t-1} = (i_1, \mathbf{Y}_1, ..., i_{t-1}, \mathbf{Y}_{t-1})$ since the reward from the action i_t will only appear after such action has been taken. Hence, what we need is a way to turn that history into an action to take, that is done by what we call the policy π_t. Our main goal is to maximize the cumulative reward over all rounds, thus maximizing $\sum_{t=1}^{n} Y_t$. The main challenge in Bandit Algorithm is that the environment, and therefore the rewards distribution, is completely unknown for the decision-maker. However, it must have some idea of the class ϵ of such environment, as the design of the policies π will be determined by such class.

2.1.1 The Regret

We now need to establish a method to evaluate the performance of the user at each time $t \leq n$. The convention is to evaluate the cumulative regret over each round.

Definition 2.1. *The regret of the decision-maker relative to a certain policy π_t over $t \leq n$ rounds is defined as the difference between the total expected reward using policy π_t and the expected reward received by the decision-maker.*

We usually calculate the regret relative to a set of policies π, which is the maximum regret relative to any policy $\pi_t \in \pi$ in the set. In other words, we measure the performance of the decision-maker relative to the optimal policy. Let's consider a stochastic bandit $\upsilon = (P_i : i \in \mathbf{K})$ from Lattimore & Szepesvári (2020) and define the regret of a Policy π_t as

$$\begin{aligned} r_n(\pi_t, \upsilon) &= n\mu^*(\upsilon) - \mathbf{E}[\sum_{t=1}^{n} \mathbf{Y}_t] \\ &= n\mu^*(\upsilon) - \mathbf{E}[\tilde{\mu}] \\ &= \mathbb{E}\Big[\sum_{t=1}^{n} \triangle_i\Big] \end{aligned} \tag{2.1}$$

The second term of the equation is the expected reward for the decision-maker, the first term is the maximum expected reward using any policy π_t with $\mu^*(\upsilon)$ defined as

$$\mu^*(\upsilon) = \max_{i \in \mathbf{K}} \int_{-\inf}^{\inf} y dP_i(y) \tag{2.2}$$

and $\triangle_i = \mu^* - \mu_i$ as the gap between the suboptimal reward and the optimal one.

From this expression we can understand one of the most important concepts in the study of Bandits, the growth rate of the regret over n. The goal is for the decision maker to achieve a sub-linear regret. Using the Bachmann-Landau notation we have that, given any set of

functions $f, g : \mathbb{N} \to [0, \text{inf})$, we define

$$f(n) = \mathcal{O}(g(n)) \leftrightarrow \limsup_{n \to \infty} \frac{f(n)}{g(n)} < \infty \tag{2.3}$$

This notation will allow us to define how regrets scale over n, as $r_n = \mathcal{O}(\sqrt{n})$ or $r_n = \mathcal{O}(log(n))$ for example. Such growth will depend on the conditions of our environment, allowing us to define the Regret Upper and Lower bound for each of our algorithms, that is, the two limits on the performance of each algorithm.

2.1.2 Stochastic Bandit Algorithms

As previously stated, in the stochastic bandit setting the decision taken by the agent and the following reward observed only depends on the distribution pay-off of each of the arms.

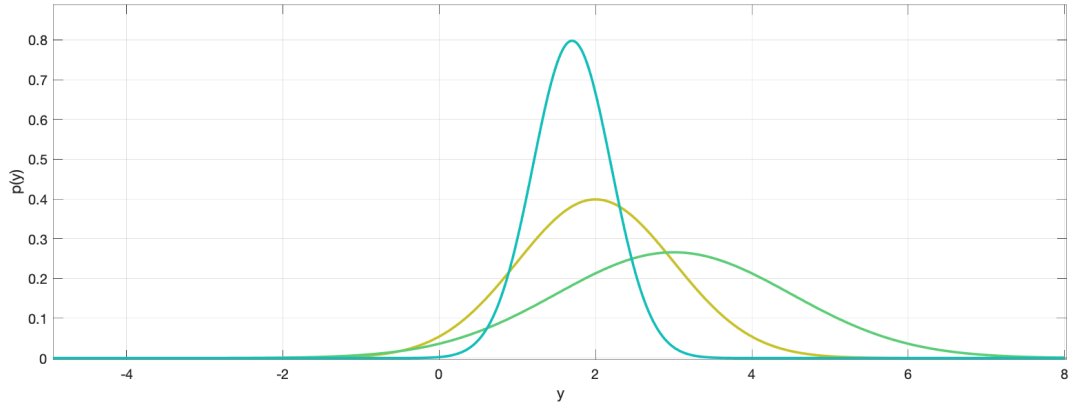


Figure 2.1: Example of normal reward distributions for a 3-armed bandit problem.

Figure 2.1 shows an example of the types of reward distributions our arms may have, generated from various Gaussians in this case. In the following sections we will introduce a few algorithms that try to learn those arm distributions to efficiently choose the best one to exploit without leaving any potential sub-optimal choice unexplored.

2.1.2.1 ϵ-greedy Algorithm

We will start by introducing the epsilon-greedy algorithm, or so called ϵ-greedy. It is the evolution of the so called *Explore First* algorithm, where the exploration phase happens uniformly at the first iterations of the algorithm, making the regret in such phase extremely high. The ϵ-greedy is the simplest case in which the exploration phase is not uniform, instead, we define what portion of the iterations we would like to explore using the parameter ϵ. The pseudo-code for the ϵ-greedy is found in Algorithm 1.

The regret bounds highly depend on the chosen value of ϵ. The best performance is found when the value of epsilon decreases exponentially for each iteration. That phenomenon is explained by the fact that exploring is more and more inaccurate as the arms have already played enough to have a close to correct mean reward for each.

Algorithm 1 ϵ-greedy Algorithm

Inputs: ϵ parameter. Time horizon T.

```
for t=1,...,T do
    Toss a coin with probability ε_t.
    if success then
        Explore. (Play an arm randomly)
    else
        Exploit. (Play the arm with the highest mean reward so far)
    end if
    μ̃_{i,t} ← (μ̃_{i,t-1} + y_t) / N_{i_t}  (Update the selected arm estimate mean.
end for
```

2.1.2.2 UCB Algorithm

In this case our goal is to estimate the mean reward of each arm using the estimates obtained from the observed rewards in previous iterations. The idea behind the Upper Confidence Bound is to define those estimates using confidence bounds for each arm that get updated every iteration where the arm is played. We will take a classical version of the UCB Algorithm introduced by Auer (2002). On any given iteration t, the algorithm chooses the action with the highest current upper confidence bound for the mean reward. This confidence bound is defined by a certain confidence parameter δ_t. As the arm is played more and more times, the confidence interval tightens around the expected mean for the arm. This gives us two scenarios in which an arm k will be played: (1) The estimate reward for such arm is the highest. (2) The arm has not been played enough times to reduce the Upper Confidence Bound. Taking into account these two cases, one should select a suitable parameter δ_t that diminishes the confidence interval fast enough to choose the optimal arm with high probability at a certain time t.

Taking all these considerations into account let's now derive the Upper Confidence Bound Algorithm. We first need to estimate the upper bound $\tilde{U}_{i,t}$ for each arm such that

$$\mu_i \leq \tilde{\mu}_{i_t,t} + \tilde{U}_{i_t,t} \tag{2.4}$$

with high probability. The width of the confidence interval depends on how many times that specific arm i has been played. Therefore, the decision maker selects the arm with the highest upper bound at each iteration as follows

$$i_t = \arg\max_{i_t \in K} \left\{ \tilde{\mu}_{i_t,t} + \tilde{U}_{i_t,t} \right\} \tag{2.5}$$

Afterwards we need to define a variable that contains the number if times an arm i has been played at each time t.

$$N_{i_t} = \sum_{j=1}^{t} \mathbf{1}(i_j = i) \tag{2.6}$$

Subsequently we can use Hoeffding's Inequality (Bentkus (2004)) to deduce the confidence Bound. The expression of the inequality applied to the bandit problem is the following

$$\mathbb{P}[\mu_i > \tilde{\mu}_{i_t,t} + \tilde{U}_{i_t,t}] \leq exp(-2N_{i_t}U_{i_t,t}^2) = p \tag{2.7}$$

We choose a probability p that exceeds the upper confidence bound to then be able to define such bound in terms of $N_{i_t,t}$ and the chosen probability p. In particular we will choose $p = t^{-4}$ which ensures asymptotically optimal arm selection (Auer (2002)).

$$U_{i_t,t} = \sqrt{\frac{-\log p}{2N_{i_t}}} = \sqrt{\frac{2\log t}{N_{i_t}}} \tag{2.8}$$

This gives us the following Upper Confidence Bound Algorithm.

Algorithm 2 Upper Confidence Bound

Inputs: p set to t^{-4}

1: **for** t=1,...,T **do**
2: $i_t \leftarrow \arg\max_{i_t \in K} \{\tilde{\mu}_{i_t,t-1} + \sqrt{\frac{2\log t}{N_{i_t}}}\}$
3: Observe reward $x_{i_t,t}$
4: **end for**

The regret analysis for the UCB Algorithm was extensively developed by Auer (2002). Let $y_i^t \in [0,1] \forall i,t$, and let $\alpha > 2$. Then the bound for the pseudo regret of the UCB algorithm is

$$\begin{aligned} R_T[UCB(\alpha)] &\leq \sum_{i:\triangle_i>0} \Big(\frac{2\alpha \ln T}{\triangle_i^2} + \frac{\alpha}{\alpha - 2}\Big)\triangle_i \\ &= \Big(\sum_{i:\triangle_i>0} \frac{2\alpha}{\triangle_i}\Big) \ln T + \frac{\alpha}{\alpha - 2} \sum_{i:\triangle_i>0} \triangle_i \end{aligned} \tag{2.9}$$

Note that the bound depends on the reward distribution Q_i via $\triangle_i$. For distribution free bounds, one can show that the regret of the α-UCB is always smaller than $\sqrt{\alpha n K \ln n}$ (Up to a numerical constant). The lower bound matches the pseudo-regret of any algorithm in the stochastic MAB setting introduced by T. Lai & Robbins (1985).

2.1.2.3 Other Algorithms for Stochastic Bandit Problems

Bayesian UCB. Until this point we haven't made assumptions on the reward distribution other than it being bounded. The Bayesian algorithms exploit prior knowledge of the rewards to then compute a probability distribution of the rewards in the next time step. That is, we try to predict the arm rewards by matching the ones we already have to a known distribution (Gaussian, Uniform...).
MOSS. This is a version of the UCB algorithm introduced by Audibert & Bubeck (2009) that stands for Minimax Optimal Strategy in the Stochastic case. It's main contribution is that it eliminates the term $\sqrt{\ln T}$ from the upper regret bound of the α-UCB algorithm through modifying the confidence level for the Upper Bound. The main drawbacks are its instability and its arbitrarily bad performance in some regimes compared to the standard UCB algorithm.

2.2 Contextual Bandits

We consider a K-armed Contextual Bandit problem, that is, a situation where the decision maker is presented with a certain context at time t and has to choose which action to take to maximize the cumulative reward. As we are considering a contextual bandit, each arm i is associated with an unknown parameter $\tilde{\beta}_i \in \mathbb{R}$ which has a dimension proportional to the Context size. That is, each component of the vector $\tilde{\beta}_i$ is associated with a component of the context vector $\mathbf{X}_t \in \mathbb{R}^d$ for arm i. We observe a different vector $\mathbf{X}_t$ when a new individual comes at each time t. As previously stated, we assume that $\{\mathbf{X}_t\}_{t\geq 0}$ is a sequence of i.i.d samples from an unknown distribution. When pulling arm $i \in [K]$, we observe a stochastic linear reward

$$Y_{i,t} = \mathbf{X}_t^\top \beta_i + \epsilon_{i,t}. \tag{2.10}$$

The noise $\epsilon_{i,t}$ is an independent σ-subgaussian random variable, that is, for all $\tau > 0$ we have $\mathbb{E}[e^{\tau\epsilon_{i,t}}] \leq e^{\tau^2\sigma^2/2}$.

As previously stated in the stochastic bandit algorithms, we will be testing the performance of our decision policy π using the cumulative expected regret, as it has been used in Bastani et al. (2017). We will compare our policy with a policy π^* that knows the true arm parameters $\{\beta_i\}_{i=1}^K \in \mathbb{R}$ in advance. We call this policy as the oracle's policy, referring to an external agent that has full information about the arm parameters prior to the experiment. Then, at any time T, the expected cumulative regret becomes

$$\begin{aligned} R_t &= \sum_{t=1}^{T} \mathbb{E}_{\mathbf{X}\sim p_X}\big[\max_{j\in[k]}(\mathbf{X}_t^\top \beta_j) - \mathbf{X}_t^\top \beta_i\big] \\ &= \sum_{t=1}^{T} \mathbb{E}_{\mathbf{X}\sim p_X}\big[\pi_t^* - \pi_t\big]. \end{aligned} \tag{2.11}$$

One can see that the above expression is simply a summation over the difference between the

reward of the decision maker and the optimal reward at each time t.

To be able to compare our results in the same environment as in Bastani et al. (2017), we need to make three assumptions on the nature of our context vector $\mathbf{X}_t$.

Assumption 1 (Parameter Set)**.** *There is a positive constant* $\mathbf{X}_{max}$ *and also a constant* b_{max} *such that*

$$\forall t \in [T] : x_{max}||\mathbf{X}_t||_2 \quad and \quad \forall i \in [K] : b_{max}||\beta_i||_2.$$

That is, the probability density px should not be defined outside the sphere of radius x_{max}*.*

The first condition basically ensures that the parameters β_i and the context vectors $\mathbf{X}_t$ are bounded. It is standard among any bandit previous work, Abbasi-Yadkori et al. (2011) for example.

Assumption 2 (Margin Condition)**.** *For* $\alpha = 1$ *There exists a constant* $C > 0$ *such that for each* $k > 0$*.*

$$\forall i \neq j : \quad \mathbb{P}_X\big[0 < |\mathbf{X}^\top(\beta_i - \beta_j)| \leq k\big] \leq Ck^\alpha = Ck$$

This second assumption sets margin condition on the observed context $\mathbf{X}_t$. We've set $\alpha = 1$ for simplicity, but any value $\alpha \geq 0$ can be also studied. As proved in Goldenshluger & Zeevi (2013), the value of α is critical to define the conversion rate on bandit algorithms, more specifically, when $\alpha = 1$ they prove that it matches bounds of $\mathcal{O}(logT)$ regret.

All the assumptions made until this point are shared among most of the bandit literature. The third assumption however, introduces a condition that guarantees that there is a diverse enough set of possible contexts under which each arm may be chosen, no matter the estimates of the arms in the previous a previous time t.

Assumption 3 (Covariate Diversity)**.** *There exists a minimum eigenvalue* λ_0 *of* $\mathbb{E}_X[XX^\top \mathbb{I}\{X^\top\mathbf{u} \geq 0\}]$ *for each vector* $\mathbf{u} \in \mathbb{R}^d$*.*

The later assumption holds for any distribution with probability density p_X bounded below a non-zero constant in an open set situated around the origin. Some examples of distribution include Uniform Distributions and truncated multivariate Gaussian distributions. A more detailed view of the later condition and the examples of distribution satisfying all three assumptions can be found in Bastani et al. (2017).

2.2.1 Greedy Bandit

Specific notation. Our context vectors $\mathbf{X}_t$ are indexed together as rows of the context matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and similarly, for $i \in [K]$ we define Y_i as the vector containing the outcomes $\mathbf{X}_t^T\beta_i + \epsilon_{i,t}$. As we can only approximate the constants β_i through least squares only

when arm i is played, we need to define a subspace containing only the entries of the context vector, outcome and idiosyncratic shock of the instances where arm i has been played. Let $\mathcal{S}_{i,t} = \{j|\pi_j = i\} \cap [t]$ be the set of times where arm i is played. Therefore, when we refer to $\mathbf{X}(\mathcal{S}_{i,t})$ we refer to the matrix containing the vectors $\mathbf{X}_t$ of the instances where arm i was played until that point.

In order to test the accuracy of the algorithm we will compare the reward of our set of policies π_t with an oracle's choice π^* that will always choose the best expected arm through $\pi_t^* = \max_{j\in[K]}(\mathbf{X}_t^\top \beta_j)$. We can then define the instantaneous expected regret as

$$r_t \equiv \mathbb{E}_{X\propto p_X}[\max_{j\in[K]}(\mathbf{X}_t^\top \beta_j) - \mathbf{X}_t^\top \beta_i]. \tag{2.12}$$

We seek to minimize the cumulative regret defined as $\mathbf{R}_T = \sum_{t=1}^{T} r_t$.

2.2.1.1 Algorithm

When a new user with context vector $\mathbf{X}_t$ comes, we use the current estimates for all arms to decide which one to play, i. e., choose which policy to take through $\pi_t = argmax_{i\in[K]}\mathbf{X}_t^T\tilde{\beta}(\mathcal{S}_{i,t-1})$. After playing arm π_t, a reward $\mathbf{Y}_{\pi_t,t} = \mathbf{X}_t^T\beta_{\pi_t} + \epsilon_{\pi_t,t}$ is observed and the current arm estimated are updated through

$$\tilde{\beta}(S_{\pi_t,t}) \leftarrow \left[\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})\right]^{-1}\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{Y}(\mathbf{S}_{\pi_t,t}) \tag{2.13}$$

It is important to note that we do not update the arm π_t estimates if $\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})$ is not invertible.

Algorithm 3 Greedy Bandit with original features

Inputs: Context vector $\mathbf{X}_t$ for all $t \in [T]$, output vector $\mathbf{Y} \in \mathbb{R}^{d\times 1}$ and finally initialize $\tilde{\beta}(S_{i,0}) = 0 \in \mathbb{R}^d$ for all $i \in [K]$.
Output: $\tilde{\beta}(\mathbf{S}_{i,T}) = 0 \in \mathbb{R}$ for all $i \in [K]$.

1: **for** $t \in [T]$ **do**
2: Check $\mathbf{X}_t \propto p_X$
3: $\pi_t \leftarrow \arg\max \mathbf{X}_i^\top \tilde{\beta}(\mathbf{S}_{i,t-1})$ (break ties randomly)
4: $\mathbf{S}_{\pi_t,t} \leftarrow \mathbf{S}_{\pi_t,t-1} \cup \{t\}$ (Include $\mathbf{X}_t$ to the subspace $\mathbf{X}(\mathbf{S}_{\pi_t,t})$)
5: Use arm π_t and get $\mathbf{Y}_{\pi_t,t} = \mathbf{X}_i^\top \tilde{\beta}_{\pi_t} + \epsilon_{\pi_t,t}$
6: **if** $\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})$ is invertible **then**
7: Update $\tilde{\beta}(\mathbf{S}_{\pi_t,t})$ by:
8: $\tilde{\beta}(\mathbf{S}_{\pi_t,t}) \leftarrow \left[\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})\right]^{-1}\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{Y}(\mathbf{S}_{\pi_t,t})$
9: **end if**
10: **end for**

2.2.1.2 Regret Analysis

The regret analysis for the Greedy Bandit was extensively developed by Bastani et al. (2017). In the case of k=2 and d=3 they prove a regret bound of

$$\begin{aligned} R_T(\pi) &\leq \frac{128C_0Cx_{\max}^4\sigma^2 d(\log d)^{3/2}}{\lambda_0^2}\log T + C\big(\frac{128C_0x_{\max}^4\sigma^2 d(\log d)^{3/2}}{\lambda_0^2} + \frac{160b_{\max}x_{\max}^3 d}{\lambda_0} + 2x_{\max}b_{\max}\big) \\ &\leq C_{GB}\log T = \mathcal{O}(\log T), \end{aligned} \tag{2.14}$$

where the constant C_0 is defined in Assumption 2 and

$$C = \big(\frac{1}{3} + \frac{7}{2}(\log d)^{-0.5} + \frac{38}{3}(\log d)^{-1} + \frac{67}{4}(\log d)^{1.5}\big) \in (1/3, 52). \tag{2.15}$$

The upper bound scales as $\mathcal{O}(d^3(\log d)^{3/2}\log T)$ in the context dimension d for the 2-armed setup. In the case of the lower bound, we use the findings from Goldenshluger & Zeevi (2013) where they establish a bound of $\mathcal{O}(\log T)$ for any two-armed contextual bandit.

The procedure to derive the regret bounds requires a deep statistical analysis of the context and the steps taken by the algorithm at every iteration. The theoretical demonstrations are out of the scope of the project.

2.2.2 Other contextual bandit algorithms (GLM-UCB and OLS)

We present the main algorithms to which the Greedy Bandit, and therefore, our modified version, can be compared to.

Generalized Linear Model Upper Confidence Bound (GLM-UCB). This algorithm is presented by Filippi et al. (2010) and it's inspired on the Upper Confidence Bound from Auer (2002). The main goal of comparing the Greedy Bandit with this algorithm is to see the performance with GLM rewards. Bastani et al. (2017) specifically compares the performance of both Algorithms with Logistic rewards, i.e. $\mathbb{Y}_{i,t}$ with probability $1/[1+\exp(-\mathbf{X}_t^T\beta_i)]$ and 0 otherwise.
The Greedy Bandit clearly outperforms the GLM-UCB algorithm in settings with both 3 and 10 features. The results can be found in Bastani et al. (2017) and allow us to state that our algorithm performs better than the GLM-UCB in the previously mentioned setup.

OLS Algorithm. The OLS Bandit algorithm was introduced by Goldenshluger & Zeevi (2013) and generalized by Bastani et al. (2019b). They present an algorithm that builds on the LASSO Estimator, converting initial high dimensional feature vectors into a reduced version with only the ones that determine the final reward. It achieves a slightly better upper bound of $\mathcal{O}(d^2(\log d)^{3/2}\log T)$ (Note from section 2.2.1.2 that only by a factor of d).

3 Relevant Machine learning techniques

3.1 Ridge regression

When our context matrix is high-dimensional, the features tend to be highly collinear. The fact that two or more features are highly linearly related causes the subspace of the covariates to not be full rank, or close to. In that situation, when our covariates matrix is rank deficient, it is almost impossible to separate the specific contribution of each feature.

Recovering our OLS formula to approximate the β parameter we have

$$\tilde{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}. \tag{3.1}$$

The strong multicollinearity we just mentioned causes $(\mathbf{X}^\top \mathbf{X})$ to be singular. In contrast, the only case in which we can define the β parameter is when $(\mathbf{X}^\top \mathbf{X})$ exists. The solution proposed by Hoerl & Kennard (1970) to fix the "almost" singularity of $(\mathbf{X}^\top \mathbf{X})$ is to add an additional term $\lambda \mathbf{I}_{p\times p}$ with $\lambda \in [0, \infty)$. The λ parameter is used to regulate the optimal contribution of the new term, the so called *ridge penalty*. Taking all of this into account we can now define the ridge regression estimator:

$$\tilde{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{p\times p})^{-1} \mathbf{X}^\top \mathbf{Y} \tag{3.2}$$

for $\lambda \in [0, \infty)$. If the value of λ is strictly positive then the estimator is well defined in all cases. It is also important to point out that each choice of λ leads to a different estimate of $\mathbf{Y}$. The set of estimators for a λ range is called the solution path of the ridge regression estimator.

3.1.1 The importance of the Lambda choice

In practice, the optimal value of the λ parameter is not given, and our goal is to find such value as accurately as possible with the least computational complexity.

In figure 3.2 we plot the test error over a range of λ values for a regression over a set of randomly generated i.i.d. data. It can be seen that the error diverges as the value of λ approaches 0, i.e. the Ridgeless limit. Then the error is kept constant for a range of small values of λ until a point where it starts decreasing. We then find a global minimum for a certain value of λ before the impact of the ridge penalty deliberately increases the test error again. That minimum indicates the optimal value of λ for which our regression model will best fit the data.

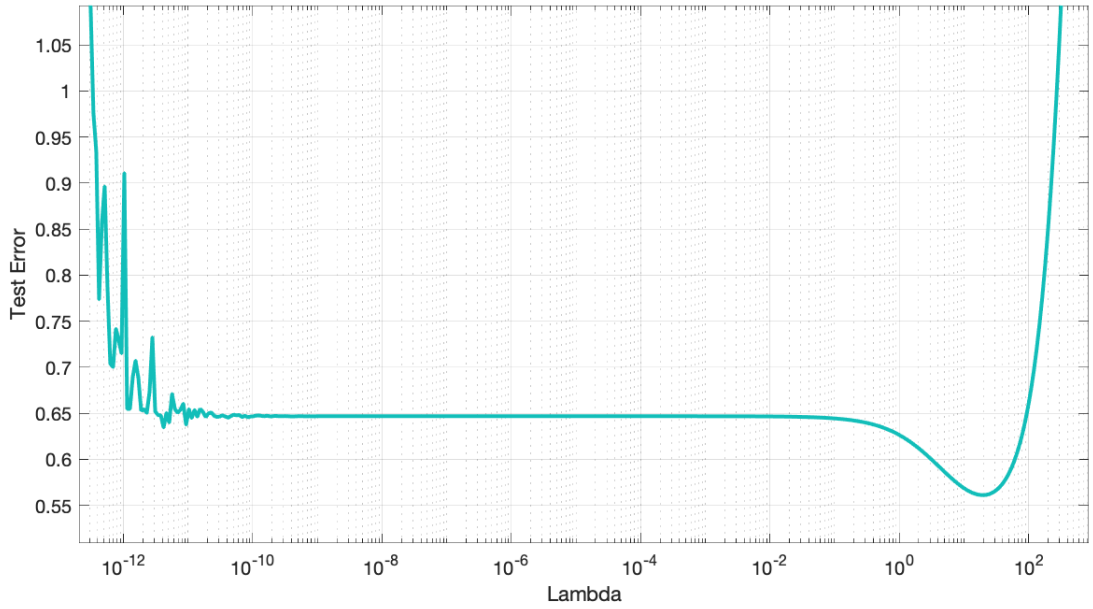


Figure 3.1: Ridge Regression over a randomly generated matrix of covariates $\mathbf{X}$ from a normal distribution $\mathbb{N}(0,1)$ and $\mathbf{Y} = \beta\mathbf{X} + \epsilon$ with ϵ as a gaussian noise with $\sigma = 10$. The figure contains the test error obtained for a range of values for the λ parameter.

There are many methods that approach the problem of finding the optimal λ for practical applications. Some of the proposed methods include cross-validation from Allen (1974), L-curve method from Hansen (2001), restricted maximum likelihood from Bartlett (1937) and others. We will focus on the cross-validation method, which is probably the most widely used nowadays.

3.1.2 Cross Validation

When looking at all the methods used to evaluate the prediction error, cross-validation may be the most commonly used in machine learning procedures. The goal is to sequentially train and test the model with different partitions of the dataset, then averaging the results and getting an unbiased final result.

Given a training data $(\mathbf{X}_i, \mathbf{Y}_i)$, $i \in \mathbb{N} \leq n$, and an estimator $\tilde{\beta}(\lambda)$ that depends on the value of λ. We discretize the chosen range of λs for which we want to run the test. In our case we have used a logarithmic range in order to cover various orders of magnitude. Then the algorithm divides the training set in k equal subsets (5 in our case). The goal is to use each of those subsets as the test set once we've trained our model with the other four sets. In algorithm 4, one can see the process to conduct a k-fold cross-validation procedure using ridge regression as the estimator and the prediction error $\|\tilde{\mathbf{X}}_{va}(\mathbf{S}_{\pi_t,t,j})\tilde{\beta}(S_{\pi_t,t,j}) - \mathbf{Y}_{va}\|^2$ as the performance measure.

Algorithm 4 Cross-validation Strategy

for $\lambda \in \mathbf{L}$ **do**
  for $j = 1 : k$ **do**
    Split $\mathbf{X}(\mathbf{S}_{\pi_t,t})$ in subgroups $\mathbf{X}(\mathbf{S}_{\pi_t,t,j})$ for all $j \in [K]$
    $\mathbf{X}_{tr}(\mathbf{S}_{\pi_t,t,j}) \leftarrow \mathbf{X}(\mathbf{S}_{\pi_t,t}) \setminus \mathbf{X}(\mathbf{S}_{\pi_t,t,j})$
    $\mathbf{X}_{va}(\mathbf{S}_{\pi_t,t,j}) \leftarrow \mathbf{X}(\mathbf{S}_{\pi_t,t,j})$
    $\tilde{\beta}(S_{\pi_t,t,j}) \leftarrow \left[\mathbf{X}_{tr}(\mathbf{S}_{\pi_t,t,j})^\top \mathbf{X}_{tr}(\mathbf{S}_{\pi_t,t,j}) + \lambda \mathbf{I}\right]^{-1} \mathbf{X}_{tr}(\mathbf{S}_{\pi_t,t,j})^\top \mathbf{Y}_{tr}(\mathbf{S}_{\pi_t,t,j})$
    Test the obtained $\tilde{\beta}(S_{\pi_t,t,j})$ in the validation set $\mathbf{X}_{va}(\mathbf{S}_{\pi_t,t,j})$ and obtain the error $\gamma_{j,k}$ by
    $\gamma_{j,k} \leftarrow \|\mathbf{X}_{va}(\mathbf{S}_{\pi_t,t,j})\tilde{\beta}(S_{\pi_t,t,j}) - \mathbf{Y}_{va}\|^2$
  end for
end for
$\lambda_t \leftarrow$ Minimum MSE in $\gamma_{j,k}$ for all $\lambda \in [K]$

Although cross-validation ensures that the analysis does not depend on the order of the observations, in cases like ours, where we need to predict a suitable lambda choice in every iteration, the process can easily become computationally non feasible after a certain number of iterations. The computational cost also depends on the sampling accuracy for our lambdas, as the algorithm has to conduct ridge regression for each of the singular values within the chosen range.

3.2 Extended Random Features

In classical statistics, it is often recommended not to use models that have too many features for a limited amount of observation in order to not "overfit" and therefore have a large testing error. The classical behavior when the number of features starts increasing keeping the number of observations fixed follows the classical U-shape:

- **Decreasing Test Error**
  As there are more and more instances that allow the model to reduce the bias as the information available to predict the parameters increases.
- **(Local) Minimum of the Test Error**
  The model achieves a local minimum where the number of features is optimal.
- **Increasing Test Error**
  The error starts peaking around the interpolation point due to the overfitting.
- **Interpolation Thereshold**
  When the training error vanishes the test error has a peak that needs to be regularized by the Lambda Value in Ridge Regression for numerical stability.

However, as we add more features, the error starts decreasing again, achieving the global minimum in the overparameterized regime. This characteristic is common in Deep Learning methods, where the neural networks often contain way more parameters than training

samples, and they still can interpolate the observed labels, even if those are created by pure noise. This phenomenon is not only present in neural networks, it has been demonstrated to appear even in linear regression, as shown in figure 3.2.

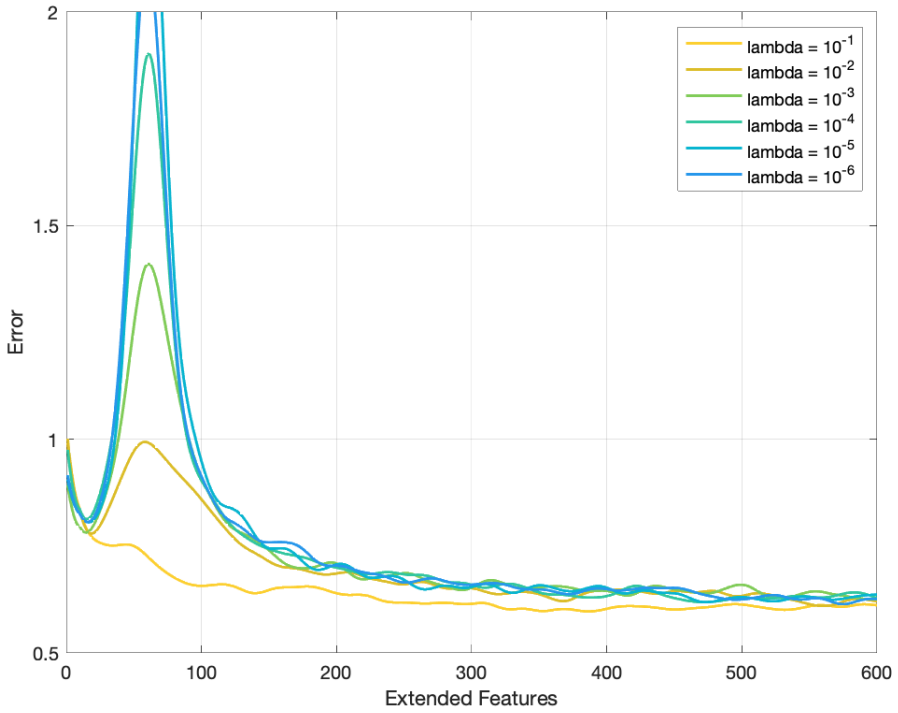


Figure 3.2: Ridge Regression over $\mathbf{X} \in \mathbb{R}^{d'}$ obtained by adding random features to the original context $\mathbf{X} \in \mathbb{R}^d$ with ReLU as the activation function ($ReLU(x) = max(0, \tilde{\mathbf{X}})$. The figure contains the plot for various values of λ that modify the stability of the system around the point where $n \equiv d$.

In our case, we want to mimic the setup in Mei & Montanari (2019) to understand the way Random Features can be generated and what specific characteristics make the phenomenon of the double descent curve appear. As clearly shown in figure 3.2 the behavior of the curve highly depends on the value of λ to control the peak around the interpolation threshold.

For our analysis we've considered the problem of learning an unknown function when given with i.i.d. samples $(\mathbf{Y}_i, \mathbf{X}_i) \in \mathbb{R} \times \mathbb{R}^d$ for all $i \leq n$. We generated the covariates $\mathbf{X}_i$ from a d-dimensional sphere of radius 1.

$$\mathbf{X}_{i \sim i.i.d.} = \text{Unif}(\mathbb{S}^{d-1}(1)) \tag{3.3}$$

Then the response is generated by

$$\mathbf{Y}_i = f_d(\mathbf{X}_i) + \epsilon_i \tag{3.4}$$

where the noise $\epsilon_i \sim i.i.d.$ and generated independently of $\mathbf{X}_i$, satisfying that $\mathbb{E}_\epsilon(\epsilon_1) = 0$ and $\mathbb{E}_\epsilon({\epsilon_1}^2) = \tau$.

We apply our extended random features transformation to our context $\mathbf{X}$ containing our entries $\mathbf{X}_i \in \mathbb{R}^d$. The transforming function will change the dimension of the new context and coefficients to $\mathbb{R}^{d'}$ with d' being the new number of features. Therefore we have

$$\mathcal{F}_{RF}(\Omega) = \left\{ f(\mathbf{X}_t; \tilde{\beta}, \Omega) \equiv \sum_{i=1}^{T} \tilde{\beta}_i \sigma\langle \mathbf{X}_t, \omega_i \rangle : \quad \tilde{\beta}_i \in \mathbb{R} \ \forall i \in [T] \right\} \tag{3.5}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the sigmoid activation function $\sigma = \frac{1}{1-e^{\mathbf{X}}}$ and the weights ω_i are i.i.d. random vectors in $\mathbb{R}^{p\times 1}$ generated from the distribution $N(0, I_{p\times p'})$. Then we learn the coefficients performing Ridge Regression

$$\tilde{\beta}(\lambda) = arg\ min \left\{ \frac{1}{n} \sum_{j=1}^{n} (\mathbf{Y}_j - \sum_{i=1}^{T} \tilde{\beta}_i \sigma\langle \mathbf{X}_t, \omega_i \rangle)^2 + N\lambda \|\tilde{\beta}_i\|_2^2 \right\}. \tag{3.6}$$

3.2.1 Activation function (σ)

Activation functions are a critical component of deep learning, as they affect the output, accuracy and computational cost of the model. They also are critical in determining the convergence speed or even the network's ability to converge in the first place.

In summary, activation functions decide weather a neuron should be activated or not and bring non-linearity to the neural network, otherwise it would essentially be a liner regression model.

In our case, we want to bring this non-linearity to our contextual bandit approach and see if we can obtain some of the properties that the neural networks offer. We analyze the following activation functions:

1. **ReLU**. The Rectified Linear Unit (ReLU) is a non-linear activation function that has gained popularity because it does not activate all neurons at the same time. It only gives a linear transformation if the output is larger than 0, otherwise the neuron is not activated at all. The expression is the following:

   $$Relu(x) = max(0, x) \tag{3.7}$$

   This is the selected activation function to generate random features for linear regression in Mei & Montanari (2019), however, in our case it has shown to have a poor performance in comparison to other functions.

2. **Sigmoid**. It is also one of the most widely used non-linear activation functions. Its expression is the following:

   $$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

As it can be easily appreciated in equation 3.8, it normalizes values from 0 to 1. It is also important to note that if the input values are higher than 1, it will group them very closely with each other. That is why in our case, we have to make sure that the $\mathbf{X}_i$ components to which we apply the activation function are somehow normalized or rescaled around the origin.

Depending on the nature of the covariates $\mathbf{X}_i$ it can be more optimal to use one or the other. For example, in the setup shown in figure 3.2, ReLU worked better than Sigmoid to show the double descent curve effect. However, as we will see in the experimental evaluation, our Extended Features Greedy Bandit Algorithm performs better with the Sigmoid activation function in most of the real datasets.

4 Extended Features Contextual Bandit

4.1 General overview of the modifications to the Greedy Bandit

As previously stated, our main objective is to apply the modifications found in the random features model from deep learning methods and the advantages of ridge penalty to make the greedy bandit approach more accurate. We've tested multiple setups, but always as a combination of the following key features:

- **Ridge Regression**
  - *Fixed Lambda*
    In this case we use a value of $\lambda \sim 10^{-9}$ with the sole purpose of getting numerical stability in the interpolation threshold.
  - *Dynamic Lambda choice through Cross Validation*
    In this case we choose a certain value of lambda for every incoming context vector $\mathbf{X}_t$, therefore finding the optimal value for each iteration. However, the main counterpart is that the process increases its computational complexity at every run, making it unfeasible in most studied models.

- **Extended Random Features**
  We apply the transformations of the context vectors using the activation function to bring the results obtained in section 3.2 to our contextual bandit setup. We use two different methods:
  - *Fixed number of Random Features*
    We define the number of extended features beforehand and we keep it fixed for all $t \in n$.
  - *Dynamically Adding Random Features*
    We sequentially add more features as the number of iterations grow. That allows us to achieve a better accuracy when the number of iterations is low.

We will analyse each of the modifications individually at first, but our ultimate goal is to end-up using an efficient combination. We will see that the efficient combination is not valid for all environments, there are certain characteristics from the context vectors that determine the impact of such modifications. Our goal is to define the conditions for each modification to be applied.

4.2 Greedy Bandit with Ridge Regression

As we have previously seen, Ridge Regression solves the problems that least squares have when multicollinearity appears, smoothing the values of the variances proportionally to the value of Lambda. In our Greedy Bandit modification we will add ridge penalty to the way we update the value of $\tilde{\beta}$ giving

$$\tilde{\beta}(S_{\pi_t,t}) \leftarrow \left[\tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})^\top \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t}) + \lambda_t \mathbf{I}\right]^{-1} \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})^\top \mathbf{Y}(\mathbf{S}_{\pi_t,t}) \tag{4.1}$$

When testing Algorithm 5 in real datasets we realize that the main contribution of having ridge penalty without dynamically finding the optimal lambda for each iteration lies on ensuring numerical stability among any given context, avoiding the inaccurate rise of the covariates around the interpolation threshold.

Algorithm 5 Greedy Bandit with Ridge Penalty

Inputs: Context vector $\mathbf{X}_t$ for all $t \in [T]$, λ value for ridge regression and, finally, $\tilde{\beta}(S_{i,0}) = 0 \in \mathbb{R}$ for all $i \in [K]$, $\mathbf{X}_t$.
Output:$\tilde{\beta}(S_{i,T}) = 0 \in \mathbb{R}$ for all $i \in [K]$.

1: **for** $t \in [T]$ **do**
2: Check $X_t \propto p_X$
3: $\pi_t \leftarrow \arg\max \mathbf{X}_i^\top \tilde{\beta}(\mathbf{S}_{i,t-1})$ (break ties randomly)
4: $\mathbf{S}_{\pi_t,t} \leftarrow \mathbf{S}_{\pi_t,t-1} \cup \{t\}$ (include $\mathbf{X}_t$ to the subspace $\mathbf{X}(\mathbf{S}_{\pi_t,t})$)
5: Use arm π_t and get $\mathbf{Y}_{\pi_t,t} = \mathbf{X}_i^\top \tilde{\beta}_{\pi_t} + \epsilon_{\pi_t,t}$
6: **if** $\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})$ is invertible **then**
7: Update $\tilde{\beta}(S_{\pi_t,t})$ by:
8: $\tilde{\beta}(S_{\pi_t,t}) \leftarrow \left[\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t}) + \lambda \mathbf{I}\right]^{-1} \mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{Y}(\mathbf{S}_{\pi_t,t})$
9: **end if**
10: **end for**

The performance of the algorithm is similar to the one seen with the Greedy Bandit algorithm. Having a lower regret bound that scales as $\mathcal{O}(\log T)$ as presented by Goldenshluger & Zeevi (2013). Defining the Upper Bound would be substantially more complex and require a thorough study out of the scope of this project.

Being able to start approximating the β_i weights from the beginning, without worrying if $\mathbf{X}(\mathbf{S}_{\pi_t,t})^\top \mathbf{X}(\mathbf{S}_{\pi_t,t})$ is invertible, allows us to have a substancially more accurate bandit for a low number of iterations in the high-dimensional features case. In chapter 5 we proof such behavior in synthetic and real datasets.

4.2.1 Greedy Bandit with Ridge Regression and dynamic λ choice through cross-validation

It is important to note that in order to select the optimal lambda in each iteration, we have to evaluate the specific context vector of such iteration to decide which lambda value to use. Moreover, when working with real datasets, one has to ensure that the previously observed context vectors are independent and not biased. To do so, we've used cross-validation in every iteration to choose the optimal lambda from a previously defined range.

The pseudo-code for the cross-validation procedure can be found in Algorithm 4. We basically use ordinary least squares for each value in the lambda range to identify where does the global minimum for test error reside. Cross-validation allows us to vary the training and test sample at every iteration multiple times, which ensures the randomness in the observations.

It is important to note that even though this would be the ideal procedure for all our calculations involving Ridge Regression, it can represent an extremely expensive computational cost if the range of λ values $\mathbf{L}$ is too large. In fact, in our case we have that the computational complexity is increased by a factor of $5 \times l$ with l as the dimension of the λ range $\mathbf{L}$.

4.3 Extended Features Greedy Bandit

The Random Features modification of the Greedy Bandit algorithm is the main contribution of the project, as it is a setup that, to our knowledge, has not been proposed yet. It considers a context vector $\mathbf{X}_t$ in $\mathbb{R}^{1\times p}$, where T is the number of iterations and p the number of features or dimensions, and a set of generated random weights $\omega \sim \mathcal{N}(0, \mathbf{I}_{p\times p'})$ where p' is the number of random features that we expect our new context vector $\tilde{\mathbf{X}}_t$ to have.

In our case we've mostly used the sigmoid activation function σ. The pseudo-code for the Extended Random Features transformation can be found in Algorithm 6.

Algorithm 6 Random Features Generation

Inputs: Context vector $\mathbf{X}_t$ for all $t \in [T]$. Dimension of the new context vector d' (Equivalent to the number of random features)

Output: Modified context vector with extended random features $\tilde{\mathbf{X}}_t$ for all $t \in [T]$.

1: $\omega \sim \mathcal{N}(0, \mathbf{I}_{p\times p'})$
2: **for** $t \in [T]$ **do**
3: Check $X_t \propto p_X$
4: $\tilde{\mathbf{X}}_t \leftarrow \sigma(\mathbf{X}_t\omega)$
With $\sigma : \mathbb{R} \to \mathbb{R}$ as the Sigmoid Activation Function $\sigma(\mathbf{X}_t) = \frac{1}{1+e^{\mathbf{X}_t}}$
Note that $\mathbf{X}_t \in \mathbb{R}^{1\times d}$ and $\tilde{\mathbf{X}}_t \in \mathbb{R}^{1\times d'}$
5: **end for**

It is also important to note that the absolute value of the features is rescaled between 0

and 1 in order to make sure that, during the transformation of the context vector, the output is not homogenized to 1 or -1.

4.4 Expected Improvements and final version of the Algorithm

All the modifications brings a different set of advantages that should be analyzed separately. Each of the advantages described below are proved in chapter 5, both for synthetic and real datasets.

Ridge Regression. Adding ridge penalty to our arms parameter estimation method has a huge impact on the datasets with high collinearity between its covariates. As previously stated, the Greedy Bandit can only choose arms randomly before it gets at least p observations for each of the arms. Ridge regression allows us to start approximating the arm parameters from the beginning, as the covariance explosion found in the interpolation threshold is smoothed by the value of λ.
Note that in the Contextual Bandit setting the algorithm gets to the interpolation threshold every time the number of observations n in which the arm i has been played is equal to the number of features p of the context vector $\mathbf{X}_t$. That is, when $\mathbf{S}_{\pi_t,t}$ becomes a $p \times p$ matrix. That means that we have an unstable point for each of our arms.

Extended Random Features. There are two main contributions of such transformation:

- *Overparameterized Regime.* (Low number of iterations and large number of features)
  This is probably the most surprising finding, an increase in accuracy when the number of features is way larger than the number of observations, even if those features are randomly generated from pure noise. The theoretical breakdown of such phenomenon has been developed by Mei & Montanari (2019).

- *Non-linear True Reward Functions.* (Low number of features and large number of iterations)
  In this case we are looking at the opposite situation, exploring how can our method improve the accuracy of the classifier when the Greedy Bandit has found the best linear approximation. What we found is that the more random features we add, the better can the algorithm fit the true reward function from the context vectors $\mathbf{X}_t$. This phenomenon is due to the Non-linear nature of the original context and also the fact that we allow the linear model to detect and fit key features that were hidden in the original context vector.

Finally, we present the pseudo-code for our Random Features Model for Contextual Multi-armed Bandits with Non-linear Reward Function in Algorithm 7, containing all the above presented modifications to the original Greedy Bandit.

Algorithm 7 Greedy Bandit with Ridge Penalty and Random Features

Inputs: Context vector $\mathbf{X}_t$ for all $t \in [T]$, vector of possible λ values $\mathbf{L}$ for ridge regression and $\tilde{\beta}(S_{i,0}) = 0 \in \mathbb{R}$ for all $i \in [K]$.

Output: Trained $\tilde{\beta}(S_{i,T}) \in \mathbb{R}$ for all $i \in [K]$, $\mathbf{Y}_t$ as the vector of outputs for regret analysis.

$\omega \sim \mathcal{N}(0, \mathbf{I}_{p \times p'})$
Initialize $\tilde{\beta}(S_{i,0}) = 0 \in \mathbb{R}^{'}$
for $t \in [T]$ **do**
  Check $X_t \propto p_X$
  $\tilde{\mathbf{X}}_t \leftarrow \sigma(\mathbf{X}_t \omega)$
  With $\sigma : \mathbb{R} \to \mathbb{R}$ as the Sigmoid Activation Function $\sigma(X) = \frac{1}{1+e^{\mathbf{X}_t}}$
  Please note that $\mathbf{X}_t \in \mathbb{R}^{1 \times d}$ and $\mathbf{X_t} \in \mathbb{R}^{1 \times d'}$
  $\pi_t \leftarrow \arg\max \tilde{\mathbf{X}}_i^\top \tilde{\beta}(\mathbf{S}_{i,t-1})$ (break ties randomly)
  $\mathbf{S}_{\pi_t,t} \leftarrow \mathbf{S}_{\pi_t,t-1} \cup \{t\}$ (Include $\tilde{\mathbf{X}}_t$ to the subspace $\tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})$)
  $\mathbf{Y}_{\pi_t,t} \leftarrow \tilde{\mathbf{X}}_i^\top \tilde{\beta}_{\pi_t} + \epsilon_{\pi_t,t}$ (Play arm π_t and get $\mathbf{Y}_{\pi_t,t}$)
  for $\lambda \in \mathbf{L}$ **do**
    for $j = 1 : k$ **do**
      Split $\tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})$ in subgroups $\tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t,j})$ for all $j \in [K]$
      $\tilde{\mathbf{X}}_{tr}(\mathbf{S}_{\pi_t,t,j}) \leftarrow \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t}) \setminus \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t,j})$
      $\tilde{\mathbf{X}}_{va}(\mathbf{S}_{\pi_t,t,j}) \leftarrow \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t,j})$
      $\tilde{\beta}(S_{\pi_t,t,j}) \leftarrow \left[\tilde{\mathbf{X}}_{tr}(\mathbf{S}_{\pi_t,t,j})^\top \tilde{\mathbf{X}}_{tr}(\mathbf{S}_{\pi_t,t,j}) + \lambda \mathbf{I}\right]^{-1} \tilde{\mathbf{X}}_{tr}(\mathbf{S}_{\pi_t,t,j})^\top \mathbf{Y}_{tr}(\mathbf{S}_{\pi_t,t,j})$
      Test the obtained $\tilde{\beta}(\mathbf{S}_{\pi_t,t,j})$ in the validation set $\tilde{\mathbf{X}}_{va}(\mathbf{S}_{\pi_t,t,j})$ and obtain the error $\gamma_{j,k}$ by
      $\gamma_{j,k} \leftarrow \|\tilde{\mathbf{X}}_{va}(\mathbf{S}_{\pi_t,t,j}) \tilde{\beta}(S_{\pi_t,t,j}) - \mathbf{Y}_{va}\|^2$
    end for
  end for
  $\lambda_t \leftarrow$ Minimum MSE in $\gamma_{j,k}$ for all $\lambda \in [K]$
  Update $\tilde{\beta}(S_{\pi_t,t})$ by:
  $\tilde{\beta}(S_{\pi_t,t}) \leftarrow \left[\tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})^\top \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t}) + \lambda_t \mathbf{I}\right]^{-1} \tilde{\mathbf{X}}(\mathbf{S}_{\pi_t,t})^\top \mathbf{Y}(\mathbf{S}_{\pi_t,t})$
end for

Tunning parameters. The only parameter to decide is the number of random features to add (p'). For high values of p' we should expect a lower regret in the initial iterations, due to the double-descent phenomena in the overparameterized regime. However, after a certain iterations the algorithm will also get to the interpolation threshold for the new number of features, losing accuracy and becoming less accurate than the Greedy Bandit. Ultimately, if the number of features were to be the same as in the Greedy Bandit, for a very high number of iterations the algorithm would find the optimal linear approximation and stop increasing the accuracy. In such settings, we should add a high number of random features, as we have empirically proven that it allows the algorithm to keep improving its approximations when the Greedy Bandit has already found its limit.

5 Results and simulations

In this chapter we will be testing the presented algorithms in Synthetic and Real Datasets.

5.1 Synthetic Data

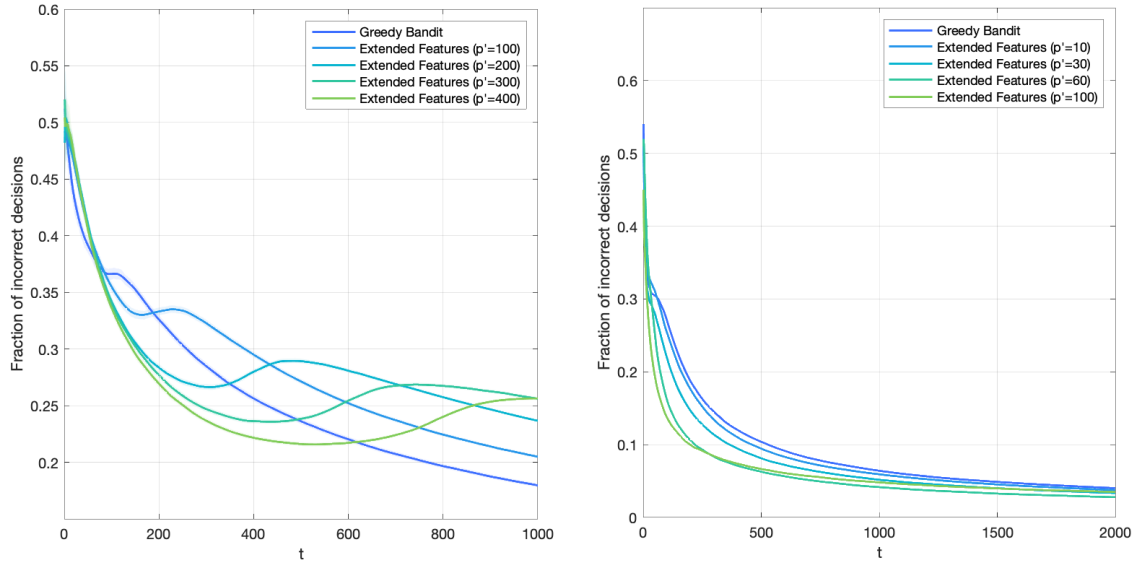
We will compare the performance of the Greedy Bandit proposed in Bastani et al. (2017) under the same conditions. Our approach allows us to directly compare the performance of our modified algorithm with the other algorithms tested in their paper. Such algorithms include the *OLS Bandit* by Goldenshluger & Zeevi (2013), which builds on ϵ-greedy methods, *OFUL* by Abbasi-Yadkori et al. (2011), which builds on the UCB method and finally, the *Prior-free* and *Prior-dependent TS* which builds on Thompson Sampling methods.

Data Generation. We need to sample our context vectors and our true arm parameters. In our case, we don't have to give any input information about the prior to our algorithm.

- **Context vectors**. We are sampling the context vectors from a truncated Gaussian distribution as
$$\mathbf{X}_t = 0.5 \times \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$$
in order to keep the l_{inf} norm bounded to being at most 1.
- We sample the arm parameters $\{\beta_i\}$ independently using a mixture of Gaussians as follows
$$\{\beta_i\} = \begin{cases} 0.5 \times \mathcal{N}(\mathbf{1}_d, \mathbf{I}_d), & \text{with } p = 0.5 \\ 0.5 \times \mathcal{N}(-\mathbf{1}_d, \mathbf{I}_d), & \text{with } p = 0.5 \end{cases}$$
- Finally, we set the noise variance to $\sigma^2 = 0.25$.

Results. In figure 5.1a and 5.1b we used the above setup with 1000 and 2000 time-steps respectively. The main difference in the setups is the number of features that our context has. In figure 5.1a we are looking at a context vector with 50 features (d=50) and only 1000 time steps. The results are plotted with a 95% confidence interval after 1000 independent iterations. It is interesting to first analyze the shape of the greedy bandit curve, which starts with a slightly better accuracy but then experiences a considerable loss of accuracy around 100 iterations. That can be explained by the fact that a k-armed bandit with 50 features would get to the overparameterization regime when each of the arms has been played 50 times.

We now analyse the extended features curves. It is straight forward to see that all of them have the same accuracy for the first 100 iterations, slightly worse than the Greedy Bandit one. If we now look at the case where $p' = 100$, we see that the loss of accuracy happens around



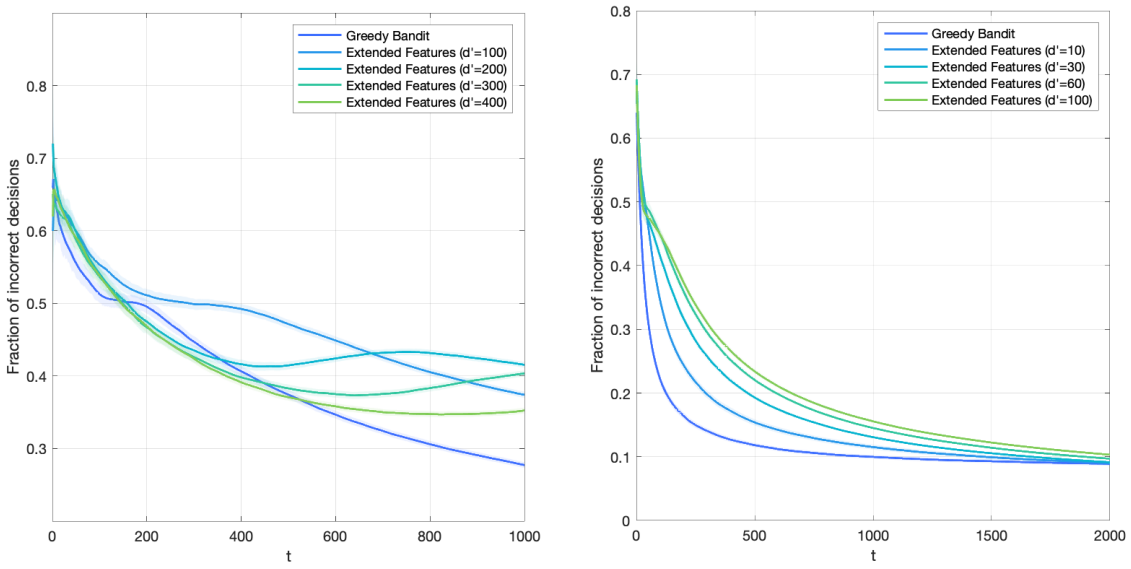
(a) Large number of features (p=50) **(b)** Small number of features (p=3)

Figure 5.1: Expected cumulative regret for our Extended Random Features Greedy Bandit with Ridge Regression algorithm using synthetic data for a 2-armed bandit. The plot has been averaged over 1000 runs showing a 95% confidence interval.

the expected point, which corresponds to the overparameterization state for each of the 2 arms. We can also see that the more we increase the extended features, the more delayed is that loss of accuracy, achieving a better accuracy for most of the experiment's iterations, proving the success of our approach. In summary, what we can observe from figure 5.1a is that in cases where we have large number of features and limited samples, one can achieve a higher accuracy using the extended features transformation than simply using the Greedy Bandit Algorithm.

In Figure 5.1b we used again a 2-armed bandit but with an original context vector that only contains 3 features. Looking at the Greedy Bandit curve, one can see that it's accuracy is way superior than the rest of the curves with the extended features model. In fact, in a very few number of iterations the algorithm already converges to the true arm parameters. The main reason for the extended features algorithm not behaving better than the Greedy Bandit is the fact that such a low number of features does not affect the curve to experiment the overparametrization covariance explosion that results in a loss of accuracy.

In Figure 5.2 we take the same values of n, p, and p' as in Figure 5.2a a and b, but in this case we are looking at a 3-armed bandit. We can easily see that in this 3-armed context the accuracy of all the algorithms is considerably reduced, specially with the Extended Features models. This phenomenon is due to the fact that we are conducting natural exploration with our Algorithms, meaning that in a setup where the number of arms is large, the algorithm may leave some arms unexploited after a certain number of iterations. We further discuss these phenomenon in 5.5b, a case where we have a 3-armed Bandit from a real healthcare



(a) Large number of features (p=50) **(b)** Small number of features (p=3)

Figure 5.2: Expected cumulative regret for our Extended Random Features Greedy Bandit with Ridge Regression algorithm using synthetic data for a 3-armed bandit. The plot has been averaged over 1000 runs showing a 95% confidence interval.

dataset. For further comparison with the previously mentioned algorithms we refer to Bastani et al. (2017), but in all the cases where the assumptions we've previously made on the context hold, the Greedy Bandit is the one that performs best.

5.2 Testing on Real Datasets

In this section we are testing the impact of the proposed algorithms in real datasets. As previously mentioned, our goal is to prove how the Extended Random Features with Ridge Regression version of the Greedy algorithm can outperform the basic version.
Before starting, we would like to mention that Bietti et al. (2018) has performed an extensive testing over more than 500 datasets in OpenML and has already shown that in more than 400 settings the Greedy algorithm outperforms the rest. That being said, we have selected 6 of those Datasets to perform a deeper analysis with our algorithm.
Five out of the six Datasets are related to Healthcare, as it has become the main purpose of the project since the COVID-19 spread around the world. The other Dataset has been chosen due to its high number of instances (88.000) and relatively low number of features (6). The 6 datasets are the following:

- **Breast Cancer Wisconsin (Diagnostic) Dataset.**
  $n = 569$, $p = 31$, $K = 2$
  The features from each entry are computed from an image of a fine needle aspirate of breast mass. The transformation of the images into a set of meaningful numerical

features was conducted using Multisurface Method-Tree (MSM-T).

- **Run or Walk Dataset**
  n = 88588, p = 6, K = 2
  This is the largest dataset used in this project, and it is used to detect if the user is walking or running. The data comes from the accelerometer and gyroscopic sensors in an Iphone 5.

The following Datasets where used in Bastani et al. (2017) and will help us compare our results with the Greedy First Algorithm.

- **EEG Eye State Dataset**
  n = 14980, p = 14, K = 2
  All the data comes from one single EEG test during 117 seconds. Such test records the brain's spontaneous electrical activity over a period of time. In this case, the purpose of such test was to detect if the eyes were open (state 1) or closed (state 2).

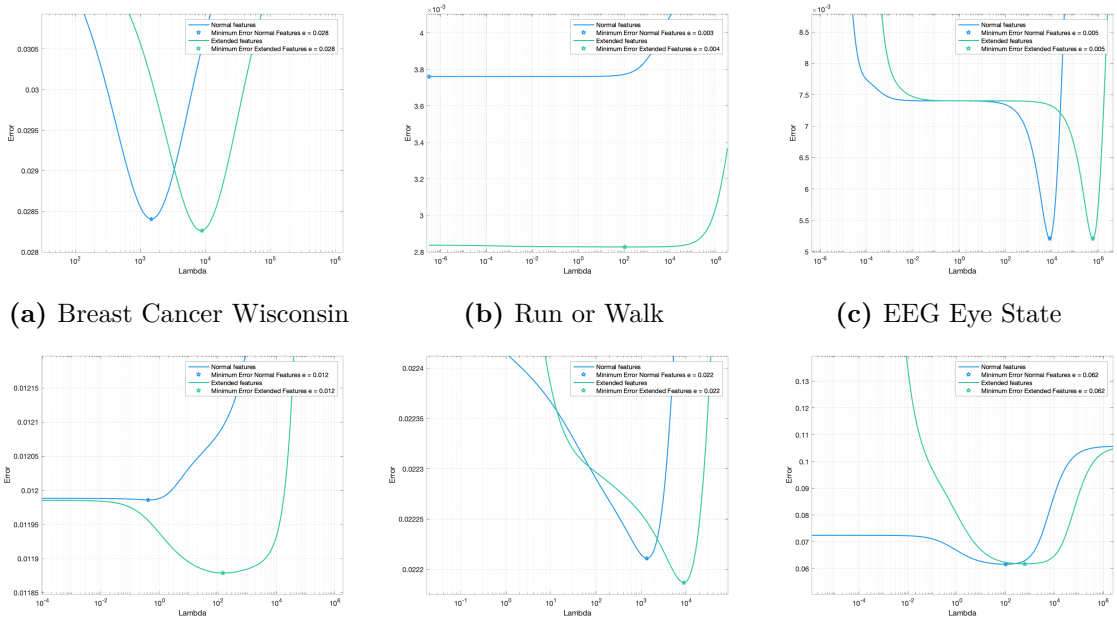
- **Eye Movements**
  n = 10936, p=22, K = 3
  In this Dataset, the decision-maker aims to classify the action of the user in three categories depending on their Eyes Movements. Thus, becoming a contextual bandit problem with 3 arms instead of 2 as in the previous Datasets.

- **Cardiotocography**
  n = 2126, p = 35, K = 3
  Processed fetal Cardiotocograms with the respective diagnostic features. We classify it in three cathegories depending on the fetal state (N, S, P).

- **Warfarin Dosage**
  n = 5528, p = 93, K = 3
  Warfarin is one of the most commonly used blood anticoagulant in the world. The problem with it is that the proper dose for each patient varies a lot. Its importance also comes from the fact that a wrong dosage can have fatal consequences for the patient.

Before testing our Bandit Algorithms on the transformed set of data, we ran a test to see if the non-linear transformation brought by our sigmoid activation function had better results in simple Ridge Regression. The steps are the following:

- Randomize the rows of our context matrix $\mathbf{X}$ in order to have i.i.d. entries.
- Split the data to get a training set and a validation one.
- Perform Ridge Regression to approximate the weights β_i over a range of λs.
- Plot the squared norm of the test error for each value of Lambda
- Repeat the process with the Extended Features version of the context vector.



(a) Breast Cancer Wisconsin **(b)** Run or Walk **(c)** EEG Eye State

(d) Eye Movements **(e)** Cardiotocography **(f)** Warfarin Dosage

Figure 5.3: Test Error after performing Ridge Regression over a logarithmic range of λ Values.

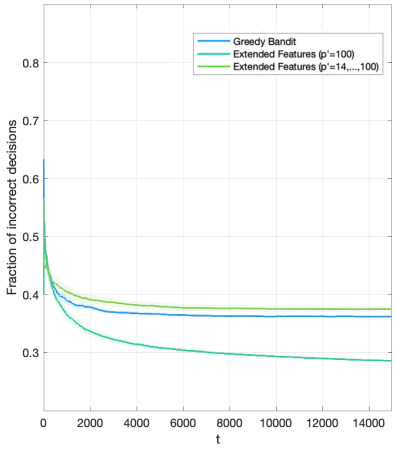
The results are plotted in figure 5.3 for all the 6 Datasets studied in this project. The goal is to see that the global minimum for the test error is found in the Extended Features Regime. That would indicate us that the data is better fit after being transformed by the sigmoid activation function combined with the random weights $w \in \mathbb{R}^{d'}$.

We can see that for most of them that is the case, specially for figure 5.3b, 5.3d and 5.3e. This means that we can better fit the data with linear regression using the randomly transformed context vector than the original one. After carrying the test, we move onto applying the actual bandit algorithms to the datasets.

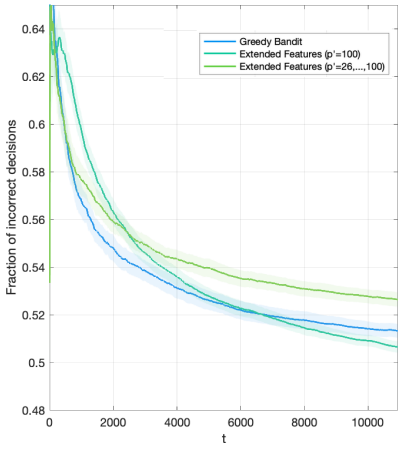
Setup. The problem becomes a classification task according to a set of features. That means, the program will classify the input context at each time t according to the current estimates for the arm parameters and will receive a binary feedback after executing the action. The reward will be 1 if the chosen class is the correct one, and 0 otherwise. Please note that as we are working with a real dataset, we will be evaluating regret rather than Bayes Regret. Due to the fact that the true arm parameters are given by real data and not simulated from prior distributions.

In figure 5.4 we mimic the setup by Bastani et al. (2017), showing the four Healthcare datasets mentioned above. In the results presented by Bastani et al. (2017), one can see a comparison with the *OFUL*, *prior-dependent TS*, *prior-free TS*, *OLS* Bandit and an Oracle's policy that knows the arm parameters in advance, which sets the limit on how good a linear model can perform.

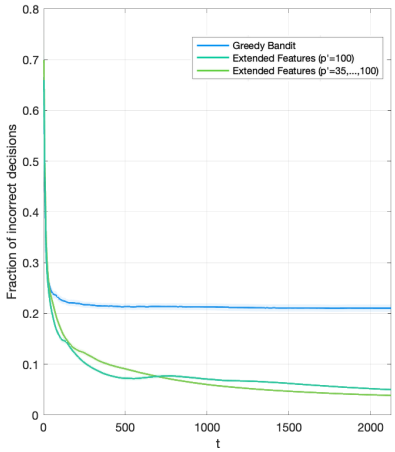
One can see a clear advantage of the Extended features version over the Greedy Bandit one



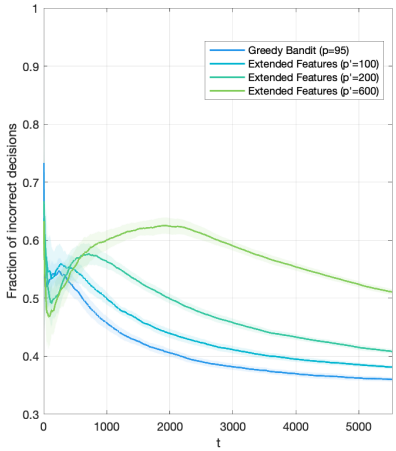
(a) EEG Eye State



(b) Eye Movements



(c) Cardiotocography

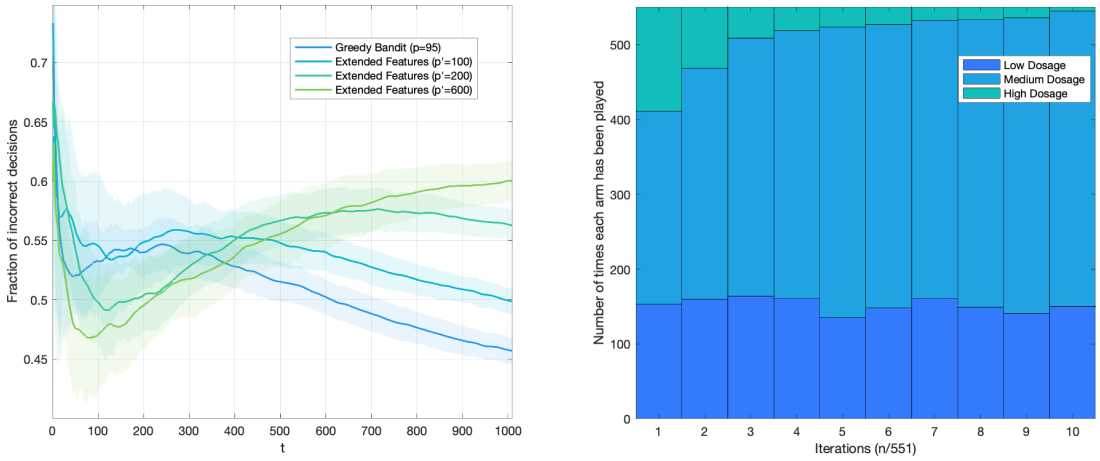


(d) Warfarin Dosage

Figure 5.4: Graphic representation of the cumulative regret achieved by the presented algorithms for the four selected healthcare datasets.

on the EEG Dataset (Figure 5.4a). It's due to the fact that the linear approximation from the Greedy Bandit can only reach the 36% average regret, because of the non-linear nature of the covariates. In fact, the Oracle's policy regret matches the one for the Greedy Bandit. In figure 5.4b we can see that although the accuracy in the first 4000 iterations is worse in the extended features version, the decreasing slope is stronger than in the Greedy Bandit, making the extended features version the most accurate one after 6000 iterations. In this case both algorithms are still far from the optimal arms true parameters, which according to the oracle's choice, would set the accuracy to 49% with the linear approximation. Figure 5.4c shows a remarkable improvement with respect to the Greedy Bandit, mostly because the substitution of OLS with Ridge Regression. In the case of the greedy bandit there are some instances in which the matrix becomes singular and therefore not invertible. That causes the difference of accuracy in this case. Finally, in figure 5.4d, we can see that for the Warfarin

Dataset the Greedy Bandit outperforms the Extended features model for most of the iterations. This case is quite special due to the little number of instances in which a certain arm is played, we take a further look into it in figure 5.5.



(a) Warfarin Dosage for 1000 instances

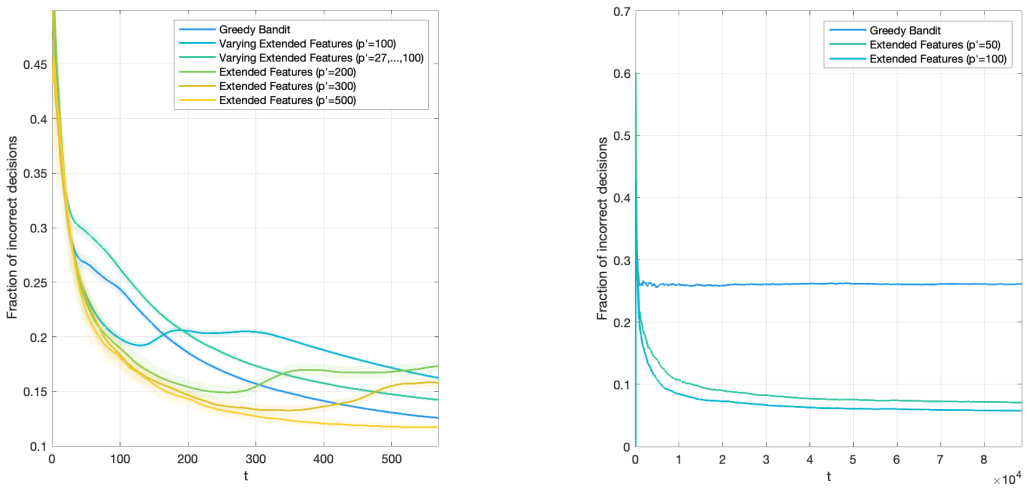
(b) Histogram of arm usage

Figure 5.5: Further study of the Warfarin Dosage dataset. Figure 5.5a shows the performance of the first iterations of both algorithms. Figure 5.5b shows an histogram of the usage of each arm over time t.

The main goal of our study is to proof a better performance in a situation where we have a lot of features and a limited amount of iterations. We've seen in Figure 5.4a, 5.4b and 5.4c that the extended features model outperforms the Greedy Bandit after many iterations, due to the better fit brought by the non-linear transformation applied to the context. However, in Figure 5.4d, we have a case in which apparently, the Extended Features model does not bring any improvement to the Greedy Bandit, as the cumulative regret after 5000 iterations keeps getting worse as we add more features. The key performance of our model in this specific case is found during the first 400 iterations, where the Extended Features Model has way more features than instances in which the arms have been updated. Figure 5.5a shows a zoomed view of the mentioned results, proving that for a limited number of iterations and a relatively high number of original features, the best performance is achieved in the overparameterized regime brought by the Extended Features model.

It is also important to analyze why even the Greedy Bandit does not achieve a good accuracy in cases like the Warfarin dataset (Figure 5.4d). In most cases, the problem lies in the fact that one or more arms end up being unused due to the limited amount of instances in which they have to be used. Therefore, even when it would be optimal to play that arm, the algorithm does not choose it. That phenomenon can be seen in figure 5.5b, where the high dosage patients stop being identified after a certain number of iterations.

Finally, we will analyze the results for the remaining datasets, one with a low number of in-



(a) Breast Cancer Diagnosis **(b)** Run or Walk dataset

Figure 5.6: Graphic representation of the cumulative regret achieved by the presented algorithms.

stances but high number of features (Breast Cancer Diagnosis) and another with an extremely large number of instances and only 6 features (Run or Walk Dataset). These Datasets were chosen because they clearly show the two main contribution of our algorithm, the advantages of overparameterization in figure 5.6a and the advantages of a nonlinear parameterization in figure 5.6b.

In figure 5.6a one can see that the more features we add, the more we displace the curve of over-fitting the arms, increasing the accuracy for low iterations. This can be used as a proof of the accuracy of the method when giving diagnosis on patients according to their sympthoms. And more importantly, it represents an improvement in the previous work when the number of patients is limited.

On the other hand, Figure 5.6b shows the results when the Algorithms are applied to a series of Iphone sensors feedback (6 in total) to determine if a person is running or walking. We can see that in this case, the more random features we add, the better is the cumulative accuracy. It proves that in such applications, a non linear transformation of the context vector can allow a better fit using linear methods afterwards.

6 Conclusions and discussion

In this work, we have explored the Greedy Bandit setting, which is proved to be the most desirable option in settings where exploration is expensive (clinical trials) and also is proven to be the most accurate in the contextual bandit setting, as shown by Bietti et al. (2018) in multiple real datasets. We also chose the Greedy Bandit because it does not have a parameter to be tuned, and therefore, can adapt to all sort of context.

We have focused on exploring what machine learning techniques could bring better performance to the greedy bandit in the situations where it has the least accuracy. The first improvement comes with the addition of ridge penalty to the OLS method to approximate the value of β. That allows us to use Greedy from the start instead of choosing the first set of arms uniformly. Moreover, adding the random features to each of our context vectors x_t allows us to take advantage of the Double Descent Curve phenomenon usually present in Deep Learning processes. In summary, ridge regression and the random features model has allowed us to radically improve accuracy, both in synthetic and real datasets, for highly dimensional covariates and a relatively low number of instances.

Another notable improvement brought by the Extended Random Features model can be seen when the Greedy Bandit achieves the best linear approximation possible. When adding more features, even though they are generated from pure noise, the algorithm is capable of interpolating more hidden features than with the original context. That phenomenon causes an increase of accuracy directly dependent on the number of random features generated. We've proven such effect for 2-armed and 3-armed context with a relatively low number of features and a high number of iterations.

All the assumptions have been tested under the same conditions as in Bastani et al. (2017) and in multiple different datasets to proof its adaptability to any set of contexts. The fact that we have shared those conditions allows the reader to compare our algorithm with the ones proposed in their paper. We've seen that our proposed method can at least achieve the Greedy Bandit accuracy, making it superior to the standard Bandit Algorithms (UCB, ϵ-greedy, Thompson Sampling...) in all the observed results.

The obtained results present a worth exploring setup for situations in which the available information is limited and we have a lot of variables to take into account. The recent COVID-19 crisis could have been one of the cases in which the proposed algorithm could have made a difference when deciding how to allocate testing at the beginning of the pandemic. The problem formulation would turn testing or not into the 2 arms to be pulled with a binary reward only received when the choice is the correct one. That situation would have involved a high-dimensional features vector, containing all the characteristics and symptoms of the patients, and would have a limited number of observations, due to the scarcity of the tests

during the first weeks of the pandemic. The only modification needed to the actual algorithm would be a dynamically restrain on how many times the "testing arm" could be pulled, as the number of tests would be limited.

The next steps for this project would be to develop a method to evaluate each specific set of context and be able to determine if they would benefit from the non-linear transformation. In our case, we've checked the non linearity of the context vector by running Ridge Regression for both algorithms over a λ range, but this could only be done because we had all the data in advance.

The conducted study opens the door to a deeper theoretical exploration focused on defining what set of conditions must the environment have and what are the expected theoretical regret bounds. It has brought new techniques to approach non-linear true reward functions with simple transformations that can have an impact on a wide set of applications.

Bibliography

Abbasi-Yadkori, Y., Pál, D., & Szepesvari, C. (2011). Improved algorithms for linear stochastic bandits. In *Nips.*

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., & Schapire, R. E. (2014). *Taming the monster: A fast and simple algorithm for contextual bandits.*

Agrawal, S., & Goyal, N. (2012). *Thompson sampling for contextual bandits with linear payoffs.*

Allen, D. M. (1974). The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, *16*(1), 125-127.

Audibert, J.-Y., & Bubeck, S. (2009, 06). Minimax policies for adversarial and stochastic bandits..

Auer, P. (2000). Using upper confidence bounds for online learning. In *Proceedings of the 41st annual symposium on foundations of computer science* (p. 270). USA: IEEE Computer Society.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, *3*, 397-422.

Ban, G., & Keskin, N. B. (2017). Personalized dynamic pricing with machine learning. *SSRN Electronic Journal.*

Bartlett, M. S. (1937, May). Properties of Sufficiency and Statistical Tests. *Proceedings of the Royal Society of London Series A*, *160*(901), 268-282. doi: 10.1098/rspa.1937.0109

Bastani, H., Bayati, M., & Khosravi, K. (2017). *Mostly exploration-free algorithms for contextual bandits.*

Bastani, H., Simchi-Levi, D., & Zhu, R. (2019a). *Meta dynamic pricing: Learning across experiments.*

Bastani, H., Simchi-Levi, D., & Zhu, R. (2019b). Meta dynamic pricing: Learning across experiments. *SSRN Electronic Journal.*

Bentkus, V. (2004). *On hoeffding's inequalities.*

Bietti, A., Agarwal, A., & Langford, J. (2018). *A contextual bandit bake-off.*

Burnetas, A. N., & Katehakis, M. N. (1997). Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, *22*(1), 222-255.

Bush, R. R., & Mosteller, F. (1953). A stochastic model with applications to learning. *The Annals of Mathematical Statistics*, *24*(4), 559–585.

Chu, W., Li, L., Reyzin, L., & Schapire, R. E. (2011). Contextual bandits with linear payoff functions. In *Aistats.*

Cohen, M., Lobel, I., & Leme, R. (2016, 07). Feature-based dynamic pricing. In (p. 817-817). doi: 10.1145/2940716.2940728

Dagan, Y., & Crammer, K. (2018). *A better resource allocation algorithm with semi-bandit feedback.*

Dani, V., Hayes, T. P., & Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *Colt.*

Filippi, S., Cappé, O., Garivier, A., & Szepesvari, C. (2010). Parametric bandits: The generalized linear case. In *Nips.*

Goldenshluger, A., & Zeevi, A. (2013, Jun). A linear response bandit problem. *Stochastic Systems*, *3*(1), 230–261. doi: 10.1287/11-ssy032

Hansen, P. (2001, 01). The l-curve and its use in the numerical treatment of inverse problems. In (Vol. 4, p. 119-142).

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55-67.

Katehakis, M. N., & Robbins, H. (1995). Sequential choice from several populations. *Proceedings of the National Academy of Sciences*, *92*(19), 8584–8585. doi: 10.1073/pnas.92.19.8584

Lai, T., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules..

Lai, T. L. (1987). Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, *15*(3), 1091–1114.

Langford, J., & Zhang, T. (2007). The epoch-greedy algorithm for multi-armed bandits with side information. In *Nips.*

Lattimore, T., & Szepesvári, C. (2020). Bandit algorithms.

Lei, H., Tewari, A., & Murphy, S. A. (2017). *An actor-critic contextual bandit algorithm for personalized mobile health interventions.*

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). *A contextual-bandit approach to personalized news article recommendation.*

Li, L., Lu, Y., & Zhou, D. (2017). *Provably optimal algorithms for generalized linear contextual bandits.*

Mei, S., & Montanari, A. (2019). *The generalization error of random features regression: Precise asymptotics and double descent curve.*

Ortner, R. (2010). Online regret bounds for markov decision processes with deterministic transitions. *Theoretical Computer Science*, *411*(29), 2684 - 2695. (Algorithmic Learning Theory (ALT 2008))

Qiang, S., & Bayati, M. (2016). Dynamic pricing with demand covariates. *SSRN Electronic Journal.* doi: 10.2139/ssrn.2765257

Russo, D., & Roy, B. V. (2013). *Learning to optimize via posterior sampling.*

Schwartz, E., Bradlow, E., & Fader, P. (2017, 04). Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, *36*. doi: 10.1287/mksc.2016.1023

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* The MIT Press.

Thompson, W. R. (1933, 12). On The Likelihood That One Unknown Probability Exceeds Another In View Of The Evidence Of Two Samples. *Biometrika*, *25*(3-4), 285-294.

Valko, M., Korda, N., Munos, R., Flaounas, I., & Cristianini, N. (2013). *Finite-time analysis of kernelised contextual bandits.*

Villar, S., Bowden, J., & Wason, J. (2015, 05). Multi-armed bandit models for the optimal design of clinical trials: Benefits and challenges. *Statistical Science*, *30*, 199-215. doi: 10.1214/14-STS504

Zhou, D., Li, L., & Gu, Q. (2019). *Neural contextual bandits with ucb-based exploration.*