Reinforcement Learning

Final Assignment

Can principal component analysis improve the effectiveness of reinforcement learning algorithms? An example from the housing loan market

March 3, 2023

Nazlıcan Eroğlu

1 Introduction

The home loan market is a multifaceted and ever-changing industry, influenced by a variety of factors that impact both the supply and demand sides. One interesting area that reinforcement learning algorithms can provide helpful insight is the decision for approval for different amounts of loans given individualistic background information. Even though banking sector aims to be transparent about the requirements as much as possible, both the demand and supply side can still benefit from a fairer and more transparent decision process given the cost of examination of the loan application from the supply side and cost of preparation of application from the demand side. We aim to compare different reinforcement learning algorithms in their maximization of approval rate given different loan categories. On top of that, we also investigate whether component analysis techniques of unsupervised machine learning literature can enhance the algorithms or not. More specifically, we aim to test whether dimensionality reduction changes the results of linear upper confidence bound (UCB) algorithm and to what extend linear UCB and Thompson Sampling (TS) algorithms are useful in maximizing approval rates.

Reducing the dimension of either the state space of the feature space is not a new topic in the reinforcement learning literature. Curran et al. (2015) argued that through the projection of the agent's state onto a low-dimensional manifold, they can create a more compact and efficient representation of the state space. Hence they can handle the issue of exponential expansion of states and actions during the exploration of optimal control in high-dimensional spaces.Bitzer et al. (2010) shows how to assess the appropriateness of dimensionality reduction as a technique for automatically identifying abstractions for reinforcement learning (RL) in robotics literature. Similar problem also reoccurs in studies regarding asset market. Guéant and Manziuk (2019) argues that strategies in asset market can typically be extended to multi-asset scenarios, solving the numerical equations that describe the optimal bid and ask quotes is infrequently addressed in the literature, particularly in high-dimensional cases. Their paper aims to introduce a numerical approach to approximate the optimal bid and ask quotes across a vast bond universe and beat the curse of dimensionality. More similar to this paper, Notsu et al. (2012) argued that reinforcement learning can be hampered by memory limitations and increased learning times due to the explosion of states. To overcome this challenge, their study employed principal component analysis as an approach to reduce the information stored in the learning table and compress the data.

This paper will proceed by providing an overview of the data, outlining the methodological background, presenting the findings, and concluding with a summary of the key results.

2 Data

The dataset, containing 614 individual data points with 12 features, is publicly available on Kaggle (Analytics Vidhya, 2016). It originates from a housing finance company and serves to automate the loan eligibility process by utilizing customer data collected from an online application form.

Among the 12 features, only four are numeric variables, namely the income of the applicant, income of the co-applicant, loan amount, and loan amount terms. The remaining features are categorical variables, including gender, employment status, number of dependents, marriage status, education level, credit history, property area, and loan status. After an initial inspection, it was found that complete data is available for only 480 individuals, so our analysis will be conducted on this reduced data. A summary of the numeric variables can be seen in the table below.

variable name	mean	variance	description
ApplicantIncome	5403	610	monthly income for applicants
LoanAmountTerm	342	65.1	loan amount terms in months
LoanAmount	146	85.5	loan amount in thousand dollars
CoapplicantIncome	1620	2920	monthly income for co-applicants

To enable the reinforcement algorithms, it is essential to define our arms and rewards. In this context, the reward represents the loan status, with a value of 1 indicating loan approval and 0 indicating otherwise. Meanwhile, the arms refer to the loan categories based on loan amounts. Prior to implementing our reinforcement learning algorithms, we analyzed the loan amounts and observed that the first 25 percent of the distribution was less than 100 dollars, while the range between the 25th percentile and the median was between 100 dollars and 128 dollars. The range from the median to the 75th percentile was between 128 dollars and 168 dollars, and the last group was above 168 dollars. Our aim was to simulate a bandit problem to maximize the loan approval rate for these arms. Proportion of loan status for each loan category can be seen in the figure 2 below.

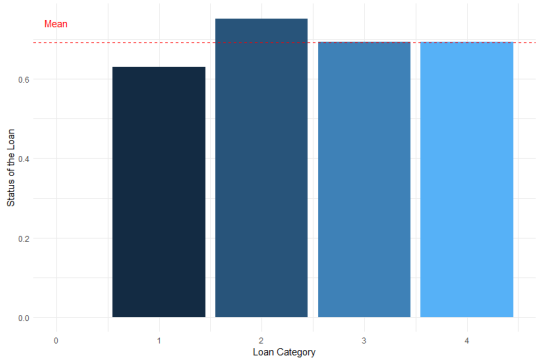

Figure 1: Proportion of Loan Status for each Loan Category

3 Methodology

To answer our research question, we will utilize TS algorithm, linear UBC algorithm with different levels of exploration and lastly factor analysis of mixed data (FAMD) (Saporta, 1990; Pagès, 2004) which is a generalized version of principal component analysis (PCA). The first two methods are popular methods in dealing with the Multi-Armed-Bandit problems.

Both TS and linear UCB methods are popular methods in dealing with the multi-armed-bandit (MAB) problems where the aim is to find the best arm/action in terms of the expected rewards. The main feature of these problems is the trade-off between exploration of new arms at the expense of exploiting a specific arm.

TS algorithm tries to deal with the problem by constructing a probability model from the obtained historical rewards for each arm. In this sense, it is different from other traditional reinforcement learning models as others rely on expected values of rewards. The algorithm works as the following: At each time step t, the algorithm samples a value from each arm's distribution and selects the arm with the highest sampled value. As more data is collected, the distributions are updated to better reflect the true underlying probabilities. It should be noted that only the selected arm distribution is updated. It is possible to demonstrate that the distributions will reach their stationary distributions, allowing us to determine the optimal arm.

More analytically, let $(A_t)_{t=1}^n$ be the action choices, $(X_t)_{t=1}^n$ the corresponding rewards and μ_i be the posterior mean of the i-th arm. Then, TS algorithm draws from each arm's distribution and chooses the arm with the largest μ_i. Then, it updates only i-th arm's distribution. It continues to draw and update until the converge condition is satisfied.

The Upper Confidence Bound (UCB), on the other hand, is another bandit algorithm that maximizes the expected cumulative reward by creating a confidence band for each arm, representing the uncertainty in the estimated reward for each action. At each time step, the arm with the highest upper confidence bound is selected. The objective function in UCB can be expressed as:

$$A_t = \underset{a}{\operatorname{argmax}} \left(Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right), \tag{1}$$

where A_t is the arm chosen, $Q_t(a)$ is the estimated value of arm a at time step t, $c \in \mathbb{R}$ is a hyperparameter that governs the degree of exploration, and $N_t(a)$ is the number of times that action a has been selected prior to time t. Linear UCB extends this loss function by assuming that there are several features which has a linear impact on the reward of each arm. It tries to solve the following optimization problem for each estimated value of arm a at time step t:

$$Q_t(a) = \underset{a}{\operatorname{argmax}} \left(x'_{a,t}(X'_{a,t-1}X_{a,t-1})^{-1}X'_{a,t-1}R_{a,t-1} + c\sqrt{x'_{a,t}(X'_{a,t-1}X_{a,t-1})^{-1}x'_{a,t}} \right), \tag{2}$$

where $X_{a,t}$ is a $N_t(a) \times m$ matrix with m features for arm a that are observed until time t, $x_{a,t}$ is a specific row of this matrix, $R_{a,t}$ is the vector that consists of the past rewards for arm a. In both algorithms, the term inside the square root represents the measure for uncertainty. Note that one big difference between TS and UCB algorithms is that in the latter the arm values are not independent as number of time that an arm is selected has an impact on the optimization.

What we propose in this paper is to use PCA to reduce the dimension of the feature space and

see whether linear UCB that uses this reduced feature space works better than the linear UCB that uses the complete feature space or not. This comparison is useful not only in cases of the curse of dimensionality but also for the cases where higher feature space makes the computation very costly or slow. If there is not a significant difference between two approaches, then in most cases it would be more beneficial to use the reduced feature space.

Since our dataset consists of both categorical and numerical variables, we decide to implement FAMD instead of PCA. The reason is that PCA cannot assign same weight to categorical and numerical variables as it tries to impose the same correlation coefficient calculation to both type of variables. It is a bad idea since the distance measure for categorical variables requires extra attention. In contrast, FAMD calculates the correlation coefficient $r^2(\mathbf{X},\mathbf{Y})$ for correlation between two numerical variables $\mathbf{Y}$ and $\mathbf{X}$ and correlation ratio $\eta^2(\mathbf{A},\mathbf{Y})$ for correlation involving a categorical variable $\mathbf{A}$. Both of these measures take a value between 0 and 1 and they are comparable with each other. More specifically, they are calculated as follows:

$$r(\mathbf{X},\mathbf{Y}) = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i-\bar{y})^2}} = \frac{cov(\mathbf{X},\mathbf{Y})}{\sigma_X\sigma_Y}, \tag{3}$$

where N is the number of rows. We also have:

$$\eta^2(\mathbf{A},\mathbf{Y}) = \frac{\sum_{m\in M_A}\sum_{k\in K_m}(\overline{y_{*,m}}-\bar{y})^2}{\sum_{m\in M_A}\sum_{k\in K_m}(\overline{y_{k,m}}-\bar{y})^2}, \tag{4}$$

where M_A is the modality of A which is equal to the number of different answers for categorical variable $\mathbf{A}$, K_m is data points taking modality m and $\overline{y_{*,m}}$ is the mean of $\mathbf{Y}$ from K_m. The calculation of these two indexes are followed by the following optimization problem:

$$C_k = \underset{\|w\|=1,\bar{w}=0}{\arg\min}\,(\sum_{j=1}^{P} r^2(w,v_j) + \sum_{j=P+1}^{P+Q}\eta^2(w,v_j)|w \perp Vect(C_1,...,C_{k-1})), \tag{5}$$

where P is the total number of numerical variables, Q is the total number of categorical variables, C_k is the unit vector centered in R^n and orthogonal to the previous principal components. K is the maximum number of components that we allow in the reduced data.

There are several points that worth mentioning before continuing to the results section. For the sake of parameter tuning, we start the analysis by comparing different degrees of exploration parameters of UCB with TS. Once we decide on the best hyperparameter c, we move to linear UCB analyses. For both of them, we will analyze the cumulative rewards and also the frequency of the choice of arms for each algorithm to comment on the fairness-to-items.

In order to ensure that FAMD results accurately represent the relative scales of the variables, it is necessary to standardize our variables. Additionally, to determine the optimal number of features, we will make use of contribution and scree plots (Cattell, 1966) for both types of variables.

For the purpose of the reinforcement algorithms, we should clearly specify our arms and reward.

The reward is the loan status, it takes the value of 1 if the loan is given to the individual and 0 otherwise. The arms are the loan categories regarding loan amounts. Before we proceed to our reinforcement learning algorithms, we have analyzed the loan amounts. We have showed that the first 25 percentage of the loan amount distribution is lower than 100 dollars, from 25th percentage to median it takes the values between 100 dollars and 128 dollars, from median to 75th percentage it is between 128 dollars and 168 dollars and last group is above 168 dollars. We aimed to simulate a bandit problem where the goal is to maximize approval rate of loans given these arms. Due to the nature of the house loan market, the approval depends not only on the amount of loan but also on several other background characteristics. That's why, we have also run linear UCB analyses with full feature space. On top of that, we have reduced the feature space by using FAMD algorithm and run the same linear UCB again. This time we have used reduced feature space. The goal of this paper is to see to what extend these results were different from each other both in terms of cumulative reward and fairness-to-items. In all of our comparisons, we have also reported TS algorithm's result for the sake of comparison.

4 Results

First, we will present our results on TS and UCB with several different exploration parameters. The reason of doing this is to pin down a exploration variable that we can also use when we compare linear UCB with full feature space and linear UCB with reduced feature space. Also, we aimed to report on TS's performance on this MAB. Given that TS do not utilize the information stemming from the feature space, we expect TS to perform worse than both types of linear UCB algorithms in cumulative reward graphs.

In figure 2, cumulative rewards for UCB algorithm with several values for exploration parameter and TS are reported with 95 percent confidence interval (CI). It is seen that all of the algorithms had a similar start. We can also see that CI for $c = 1$ is the largest at all times. This was expected because it is the algorithm that spends most time on exploring other arms which makes the reward accumulation noisier. Overall, we can argue that TS algorithm does a better job on the long-run compared to other algorithms. In the later rounds, we see that high exploration comes with a lower cumulative reward for $c = 1$ and also increase in the CI. However, we also see that the CI shrinks more and more for $c = 0.1$.

When we look at the number of selections for each arm among different algorithms in figure 3, we see that only TS and UCB with $c = 1$ have explored all of the arms even though this exploration is more homogeneous between arms for TS. Overall, TS is doing a better job in both aspects. Both of these figures suggest us to use $c = 1$ for the linear UCB analyses as it explores more similar to TS and it also follows a similar path to TS in cumulative reward graph.

Before we move to the linear UCB algorithm, we run the FAMD algorithm to reduce our feature space. Figure 4 illustrates the impact of each variable on explaining the variance of the entire

dataset. The left side shows the numeric variables, with the length of each vector corresponding to its impact. On the right side, the categorical variables are depicted. Since the distance between the answers for each category is not as interpretable as it is with numeric variables, the vector representation is not used here. Instead, the relative importance of each answer for the categorical variables is shown. For instance, being female has a much greater contribution to explaining the variance compared to being a graduate. Figure 5 provides a clearer picture. It shows that loan amount, marriage status, dependent number, income, and gender contribute more to the first dimension than the average contribution indicated by the red dotted line. For the second dimension, only loan status and credit history are significant.

Overall, the FAMD results reveal that this data can be explained by 4 or 5 features, with the following being strong candidates: loan amount, applicant income, marriage status, gender, and dependents. Furthermore, we conducted a correlation check among several variables, with more detailed results available in the appendix. Additionally, the appendix includes figures demonstrating how gender and marriage categorize individuals in this dataset. Given these results, we proceed to linear UCB analyses with the following features: gender, dependents, marriage status and applicant income.

In figure 6, the cumulative rewards for linear UCB with complete feature space, reduced feature space and TS algorithms are reported. As stated before, we have set the exploration parameter to 1. The results for $c = 0.1$ can also be seen in the appendix. It is seen that all of them perform really similar except the fact that TS algorithm has the widest CI. In figure 7, number of selections for each arm is for all the algorithms are reported. It is seen that both of the linear UCB algorithms give more weight to exploration than TS algorithm. Nevertheless, we cannot argue that they produce very different results. In the light of these, we can conclude that reducing the feature space via FAMD will not change the results and it would be recommended to use such unsupervised machine learning techniques to lower the computational cost/time.

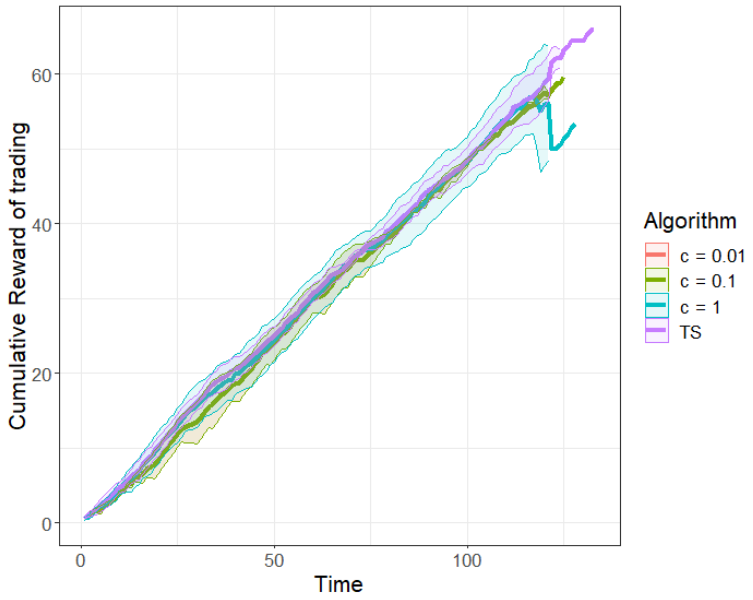

Figure 2: Cumulative Reward for UCB for alpha=1, 0.1, 0.01 and TS algorithm

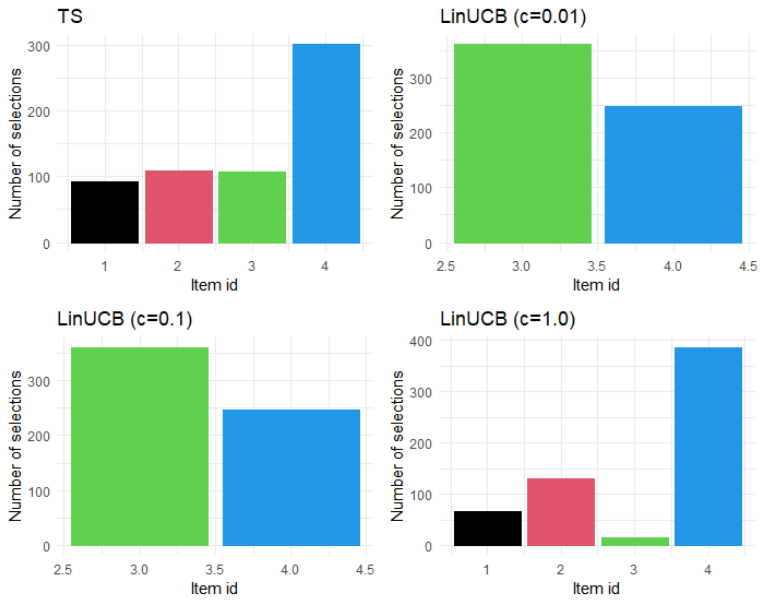

Figure 3: Selected arm histograms for UCB for alpha=1, 0.1, 0.01 and TS algorithm

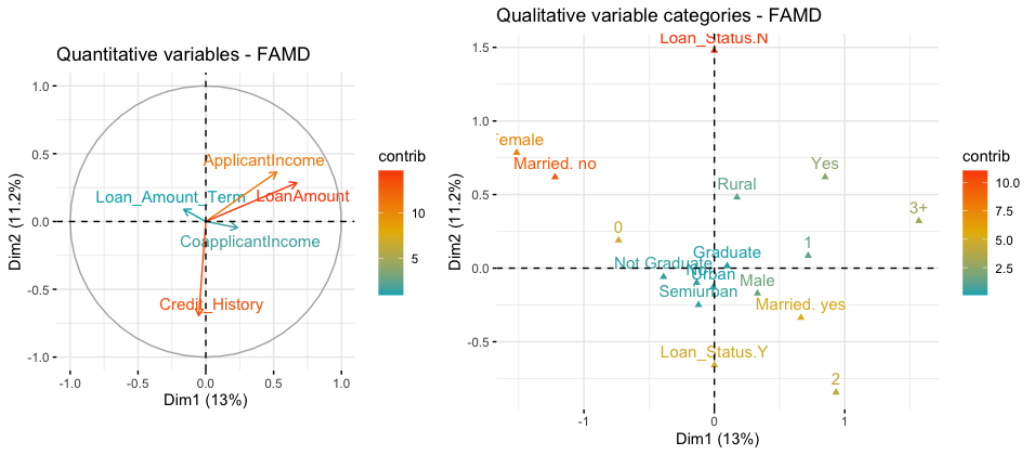

Figure 4: Left-side: Numeric variables' quality of representation on the factor map for FAMD
Right-side: Categorical variables' quality of representation on the factor map for FAMD

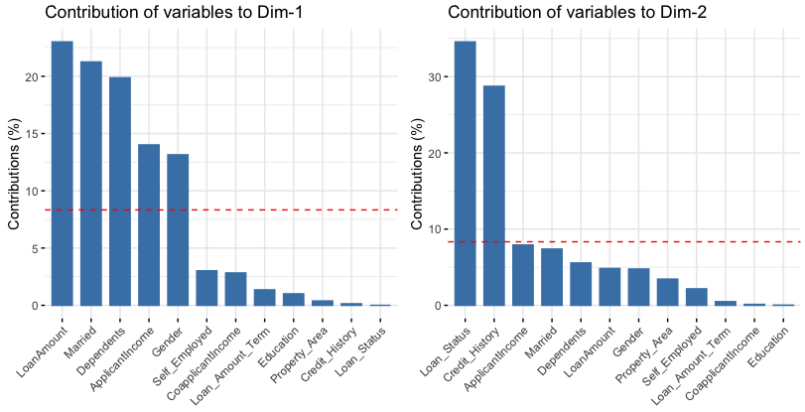

Figure 5: Left-side:Variables' contribution to the first dimension
Right-side: Variables' contribution to the second dimension

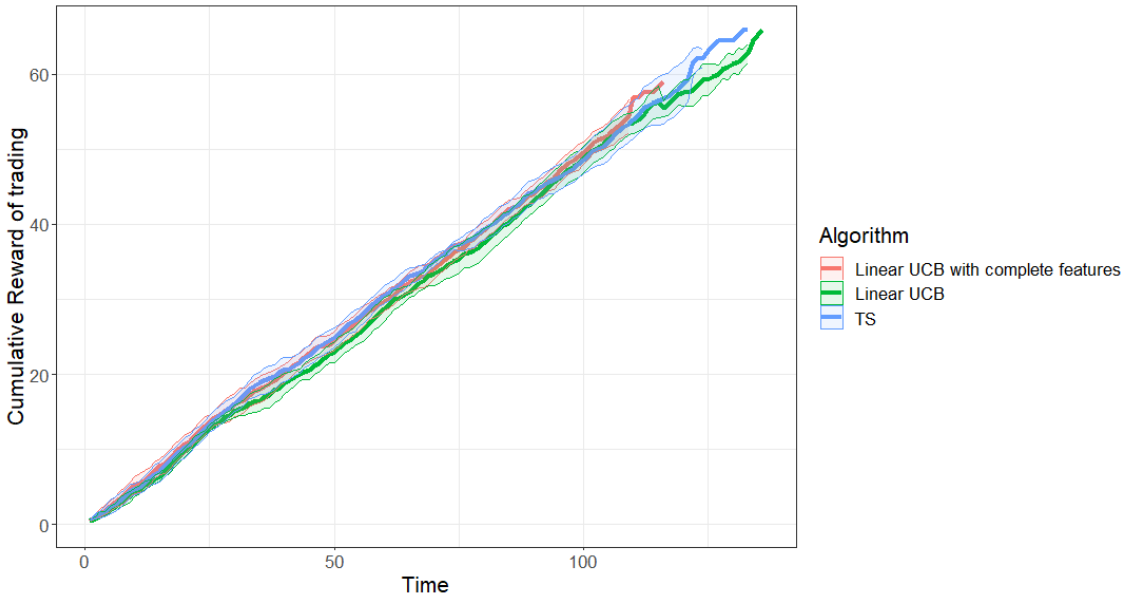

Figure 6: Cumulative Reward for Linear UCB (alpha=1) with complete feature space, features selected by FAMD and TS algorithm

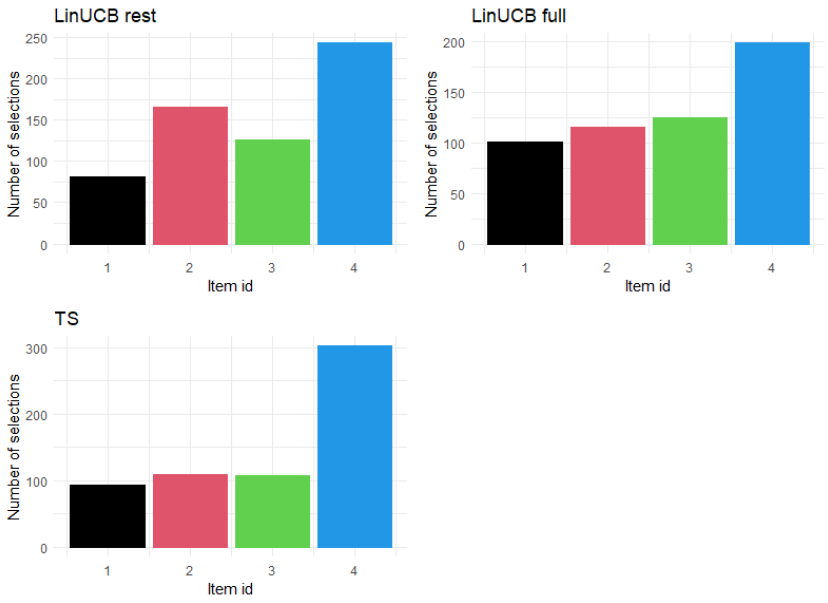

Figure 7: Selected arm histograms for Linear UCB (alpha=1) with complete feature space, features selected by FAMD and TS algorithm

5 Conclusion

In conclusion, our analysis has demonstrated that both types of linear UCB and TS algorithms produce better results in terms of cumulative reward compared the regular UCB. We also showed

that linear UCB algorithms and TS algorithm behaves similar when it comes to fairness-to-items. We are aware that this cannot be generalized for any MAB problem that a credit or a loan market could face. However, in the light of this and previous other similar studies, it is recommended to utilize unsupervised machine learning techniques alongside reinforcement learning techniques to lower the technical costs.

6 Appendix

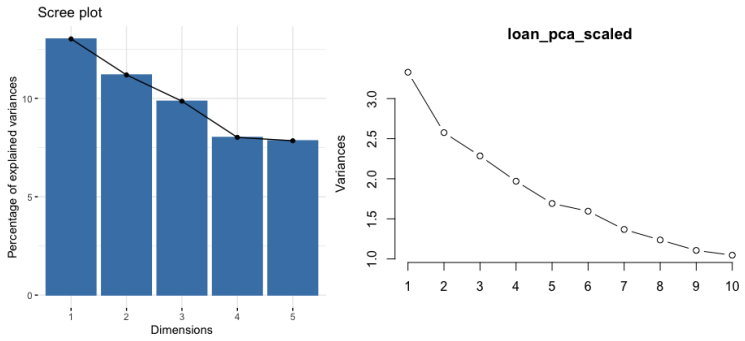

Figure 8: Left-side: Scree plot for FAMD
Right-side: Scree plot for PCA on the scaled data

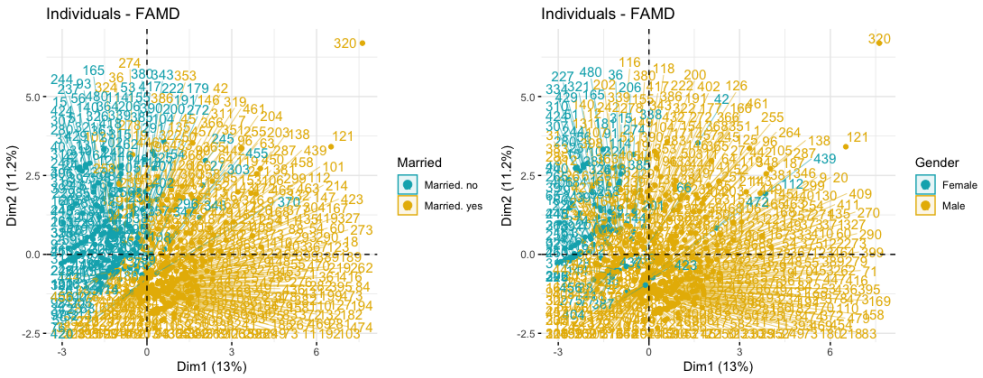

Figure 9: Left-side: graph for individuals with similar profiles in marriage status
Right-side: graph for individuals with similar profiles in gender

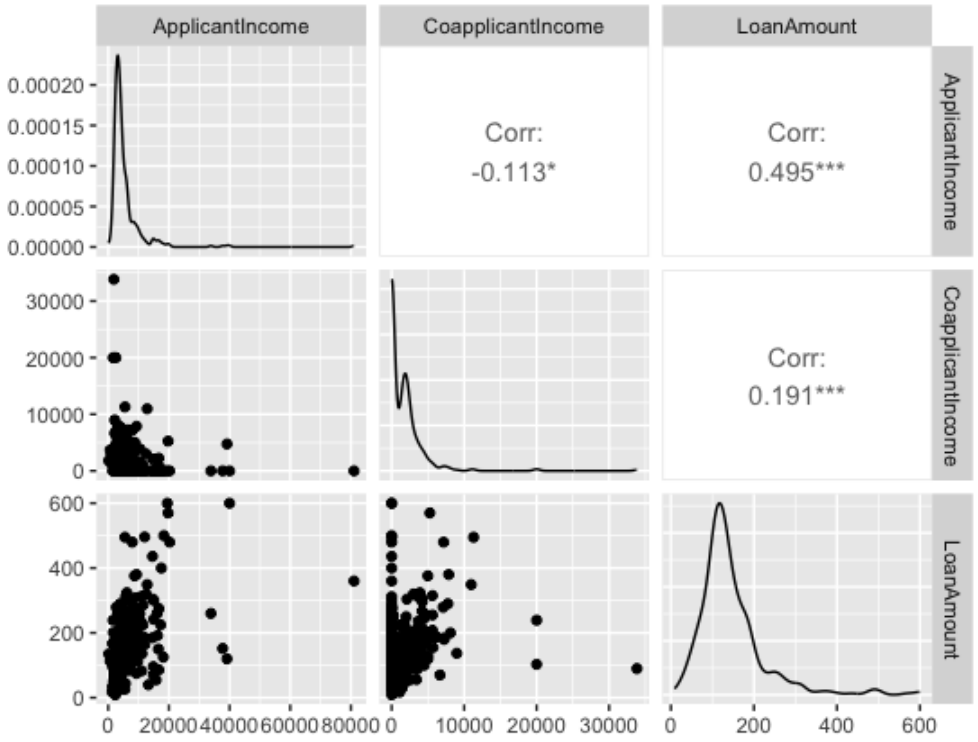

Figure 10: Scatterplot matrix for some of the numeric variables

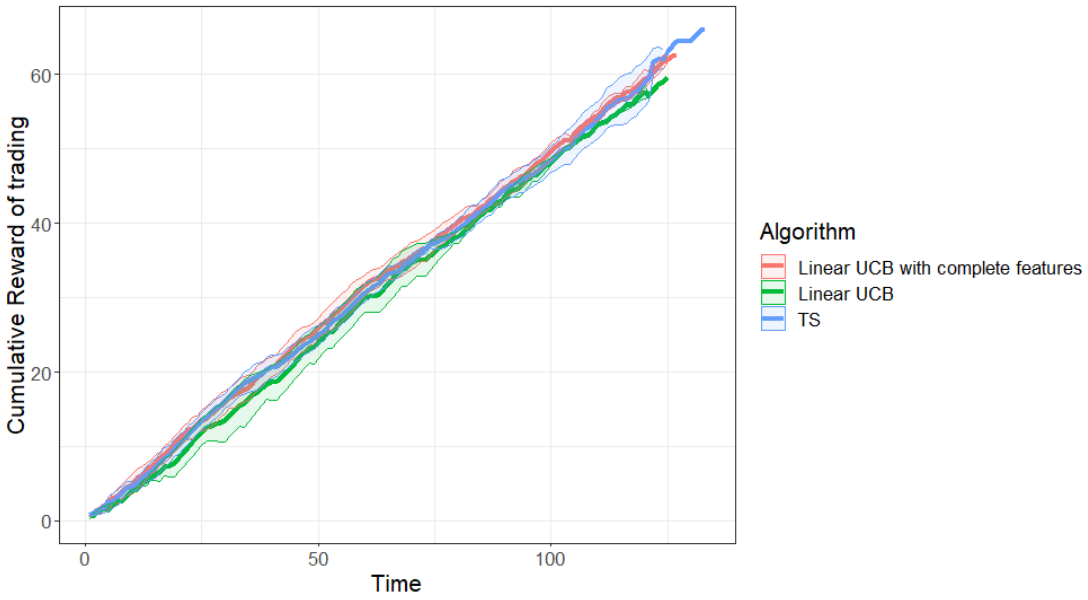

Figure 11: Cumulative Reward for Linear UCB (alpha=0.1) with complete feature space, features selected by FAMD and TS algorithm

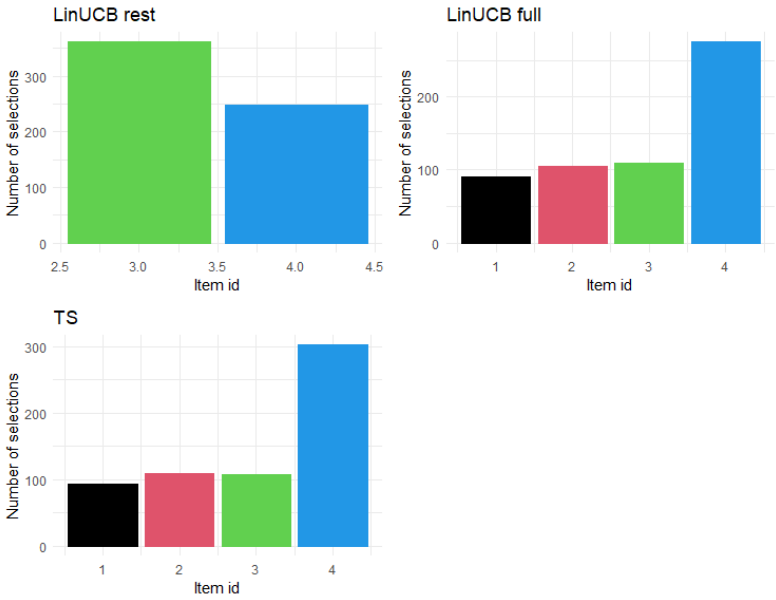

Figure 12: Selected arm histograms for Linear UCB (alpha=0.1) with complete feature space, features selected by FAMD and TS algorithm

References

Analytics Vidhya, A. (2016). Loan prediction dataset.

Bitzer, S., M. Howard, and S. Vijayakumar (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3219–3225. IEEE.

Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate behavioral research 1*(2), 245–276.

Curran, W., T. Brys, M. Taylor, and W. Smart (2015). Using pca to efficiently represent state spaces. *arXiv preprint arXiv:1505.00322*.

Guéant, O. and I. Manziuk (2019). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance 26*(5), 387–452.

Notsu, A., K. Honda, H. Ichihashi, A. Ido, and Y. Komori (2012). Information compression effect based on pca for reinforcement learning agents' communication. In *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, pp. 1318–1321.

Pagès, J. (2004). Analyse factorielle de donnees mixtes: principe et exemple d'application. *Revue de statistique appliquée 52*(4), 93–111.

Saporta, G. (1990). Simultaneous analysis of qualitative and quantitative data. In *Societa Italiana di Statistica. XXXV riunione scientifica*, Volume 1, pp. 62–72. CEDAM.

7 Code

7.1 Code

Listing 1: SML code for Final Assignment

```
### file path etc
#rm(list=ls())
options(scipen=6, digits=4)
##packages anf libraries
if (!require("pacman")) install.packages("pacman")
pacman::p_load(ggplot2,contextual, tidyverse, tinytex, rmarkdown, glmnet, matlib, MASS,
    pdist, PMA, softImpute, dplyr,plotrix, kernlab,ranger, randomForest, knitr,
               ggplot2,
               png,
               tidyverse,
               rlist,
               contextual,
               lubridate,
               zoo,
               roll)
install.packages("ggplot2")
install.packages("FactoMineR")
install.packages("vcd")
install.packages("fpc")
install.packages("roll")
install.packages("factoextra")
install.packages("zoo")
install.packages("fastDummies")
install.packages("GGally")
install.packages("cluster")
install.packages("NbClust")
install.packages("magrittr")
install.packages("data.table")
install.packages("tidyverse")
install.packages("cowplot")
library(data.table)
library(ggplot2)
library(tidyverse)
library(cowplot)
library(magrittr)
library(GGally)
library(fastDummies)
library(dplyr)
library(matlib)
library(glmnet, quietly = TRUE)
library(caTools)
library("PMA")
library("softImpute")
library(FactoMineR)
library(vcd)
library(factoextra)
library(cluster)
library(NbClust)
library(zoo)
library(readr)
library(contextual)

##Data & seed
set.seed(8913)
data <- read_csv("~/Desktop/USML/Loan data for finak/loan_sanction_train.csv")
loan_data<-data[,c(2,3,4,5,6,7,8,9,10,11,12,13)]
View(loan_data)

#Check whether we have omitted variables
```

```r
summary(loan_data)
sum(complete.cases(loan_data))
loan_data_final<-loan_data[complete.cases(loan_data), ]
str(loan_data_final)
sum(complete.cases(loan_data_final))


#Organising and scaling
loan_data_final$Gender <- as.factor(loan_data_final$Gender)  # Convert character column to factor
loan_data_final$Married <- as.factor(loan_data_final$Married)  # Convert character column to factor
loan_data_final$Dependents <- as.factor(loan_data_final$Dependents)  # Convert character column to factor
loan_data_final$Education <- as.factor(loan_data_final$Education)  # Convert character column to factor
loan_data_final$Self_Employed <- as.factor(loan_data_final$Self_Employed)  # Convert character column to factor
loan_data_final$Property_Area <- as.factor(loan_data_final$Property_Area)  # Convert character column to factor
loan_data_final$Loan_Status <- as.factor(loan_data_final$Loan_Status)  # Convert character column to factor

loan_data_final[, 6:10] <- scale(loan_data_final[, 6:10]) #Scale numeric variables

loan_data_final$Married<-tolower(loan_data_final$Married) #this is done to prevent an error in the FAMD graphs
loan_data_final$Married<- paste("Married_", loan_data_final$Married)

#PCA data only works with numeric. So we transform categorical variables into dummies
loan_pca_data <- dummy_cols(loan_data_final)
loan_pca_data<-loan_pca_data[,c(6,7,8,9,10,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29)]


#Factor Analysis for Mixed Data
loan_famd <- FAMD(loan_data_final, graph=TRUE)
summary(loan_famd)
fviz_screeplot(loan_famd,choice = "eigenvalue")
fviz_famd_var(loan_famd, repel = TRUE)
fviz_contrib(loan_famd, "var", axes = 1)
fviz_contrib(loan_famd, "var", axes = 2)

#graph of qualitative variables
fviz_famd_var(loan_famd, "quanti.var", repel = TRUE,
              col.var = "black")
fviz_famd_var(loan_famd, "quanti.var", col.var = "contrib",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE)

#graph of qualitative variables
quali.var <- get_famd_var(loan_famd, "quali.var")
quali.var

fviz_famd_var(loan_famd, "quali.var", col.var = "contrib",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
)

ind <- get_famd_ind(loan_famd)
ind

fviz_famd_ind(loan_famd, col.ind = "cos2",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE)

fviz_mfa_ind(loan_famd,
```

```
              habillage = "Gender", # color by groups
              palette = c("#00AFBB", "#E7B800", "#FC4E07"),
              addEllipses = TRUE, ellipse.type = "confidence",
              repel = TRUE # Avoid text overlapping
)

fviz_mfa_ind(loan_famd,
              habillage = "Married", # color by groups
              palette = c("#00AFBB", "#E7B800", "#FC4E07"),
              addEllipses = TRUE, ellipse.type = "confidence",
              repel = TRUE # Avoid text overlapping
)


fviz_mfa_ind(loan_famd,
              habillage = "Dependents", # color by groups
              palette = c("#00AFBB", "#E7B800", "#FC4E07","green"),
              addEllipses = TRUE, ellipse.type = "confidence",
              repel = TRUE # Avoid text overlapping
)

#PCA (here we use data we have organized for PCA)
loan_pca <- prcomp(loan_pca_data)
## Scree plot of component variances
plot(loan_pca, type = "l")

## Biplot showing the first two components and the variable loadings for these
biplot(loan_pca)

## Redo but know with columns normalized
loan_pca_scaled <- prcomp(loan_pca_data, scale = TRUE)
loan_pca_scaled

## Inspect the results as before
summary(loan_pca_scaled)
plot(loan_pca_scaled, type = "l")
biplot(loan_pca_scaled)
## Result: keep 4 or 5 variables: loan amount, married, dependents, income and maybe
    gender

######REINFORCEMENT
##Data & seed
set.seed(8913)
loan_data<- read_csv("loan_data_set.csv")
View(loan_data)
attach(loan_data)

#Check whether we have omitted variables
summary(loan_data)
sum(complete.cases(loan_data))

#create a category
summary(loan_data$LoanAmount)

loan_data$loancat <- NA

loan_data$loancat[LoanAmount >  168.0 ] <- 1
loan_data$loancat[LoanAmount >  128.0 & LoanAmount <= 168.0] <- 2
loan_data$loancat[LoanAmount >  100.0 & LoanAmount <=  128.0] <- 3
loan_data$loancat[LoanAmount <=  100.0] <- 4

loan_data_final<-loan_data[complete.cases(loan_data), ]
str(loan_data_final)
sum(complete.cases(loan_data_final))
loan_data_final <- mutate(loan_data_final, Loan_Status2 = ifelse(Loan_Status=="Y",1,0))
loan_data_final <- mutate(loan_data_final,  Married2 = ifelse(Married=="Yes",1,0))
```

```r
loan_data_final <- mutate(loan_data_final,  Dependents2 = ifelse(Dependents=="3+",1,0))
loan_data_final$Dependents[loan_data_final$Dependents=="3+"]<- 3
loan_data_final$Dependents <- as.numeric(loan_data_final$Dependents)
loan_data_final$Credit_History <- as.numeric(loan_data_final$Credit_History)
loan_data_final <- mutate(loan_data_final,  Male = ifelse(Gender=="Male",1,0))
loan_data_final <- mutate(loan_data_final,  Edu = ifelse(Education=="Graduate",1,0))
loan_data_final <- mutate(loan_data_final,  Self_E = ifelse(Self_Employed=="Yes",1,0))

attach(loan_data_final)
##Some pre-analysis graphs

table1 <- table(loan_data_final$loancat, loan_data_final$Loan_Status2)
table1_prop <- prop.table(table1, margin = 1)
table1_prop <- cbind(sort(unique(loan_data_final$loancat)), table1_prop[,2])
colnames(table1_prop) <- c("loan_cat", "status")
table1_prop <- data.frame(table1_prop)

table1plot <- ggplot(table1_prop, aes(x=loan_cat, y=status, fill=as.numeric(loan_cat))) +
geom_bar(stat="identity", position=position_dodge(), show.legend =
        FALSE)+
labs(title="", x="Loan Category", y = "Status of the Loan")+
 geom_hline(yintercept=mean(table1_prop$status), linetype="dashed",
           color = "red")+
annotate("text", x = 0, y = mean(table1_prop$status)+0.05, label =
        "Mean", color = 'red')+
theme_minimal()


##############Thompson
ts_bandit<- OfflineReplayEvaluatorBandit$new(formula = Loan_Status2 ~ loancat , data =
    loan_data_final, randomize = FALSE)
agents  <- list(Agent$new(LinUCBDisjointOptimizedPolicy$new(0.01),  ts_bandit, "c = 0.01")
    ,
                Agent$new(LinUCBDisjointOptimizedPolicy$new(0.1),  ts_bandit, "c = 0.1"),
                Agent$new(LinUCBDisjointOptimizedPolicy$new(1.0),  ts_bandit, "c = 1.0"),
                Agent$new(ThompsonSamplingPolicy$new(alpha = 1, beta = 1), ts_bandit, "TS"
    ))


# simulate
size_sim=100000
n_sim=12
newsimulation <- Simulator$new(agents, horizon = size_sim, simulations = n_sim)

ts_history           <- newsimulation$run()


# gather results
df_ts_result <- ts_history$data%>%
  select(t, sim, choice, reward, agent)


# check how many obs
max_obs_per_sim_ts = df_ts_result%>%
  group_by(sim, agent) %>%
  summarise(max_t = max(t))
max_obs_per_sim_ts

# Maximum number of observations
max_obs=1700

# data.frame aggregated for two versions: 20 and 40 arms
df_coins_agg_cumulative <- df_ts_result %>%
  group_by(agent, sim)%>% # group by number of arms, the sim
  mutate(cumulative_reward = cumsum(log(1 + reward)),
         rolling_reward = rollmean( reward, 252, na.pad=TRUE))%>%
```

```
  group_by(agent, t) %>% # group by number of arms, the t
  summarise(avg_cumulative_reward = mean(cumulative_reward),
            avg_rolling_reward = mean(rolling_reward),
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim),
            se_rolling_reward = sd(rolling_reward, na.rm=TRUE)/sqrt(n_sim)) %>%
  mutate(cumulative_reward_lower_CI =avg_cumulative_reward - 1.96*se_cumulative_reward,
         cumulative_reward_upper_CI =avg_cumulative_reward + 1.96*se_cumulative_reward,
         rolling_reward_lower_CI =avg_rolling_reward - 1.96*se_rolling_reward,
         rolling_reward_upper_CI =avg_rolling_reward + 1.96*se_rolling_reward)%>%
  filter(t <=max_obs)



# create ggplot object
ggplot(data=df_coins_agg_cumulative, aes(x=t, y=avg_cumulative_reward, color =agent))+
  geom_line(size=1.5)+
  geom_ribbon(aes(ymin=cumulative_reward_lower_CI ,
                  ymax=cumulative_reward_upper_CI,
                  fill = agent,
  ),
  alpha=0.1)+
  labs(x = 'Time', y='Cumulative Reward of trading', color ='Algorithm', fill='Algorithm')
    +
  theme_bw()+
  theme(text = element_text(size=16))

# plot number of selections
# gather results
df_TS <- df_ts_result %>%filter(agent=='TS')
df_UCB001 <- df_ts_result %>%filter(agent=='c = 0.01')
df_UCB01 <- df_ts_result %>%filter(agent=='c = 0.1')
df_UCB1 <- df_ts_result %>%filter(agent=='c = 1.0')

p001 <- ggplot(df_UCB001, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_UCB001$choice))),
           fill = as.numeric(sort(unique(df_UCB001$choice))))+
  labs(title="LinUCB (c=0.01)", x="Item id", y = 'Number of selections')+
  theme_minimal()
p001


p01 <- ggplot(df_UCB01, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_UCB01$choice))),
           fill = as.numeric(sort(unique(df_UCB01$choice))))+
  labs(title="LinUCB (c=0.1)", x="Item id", y = 'Number of selections')+
  theme_minimal()
p01

p1 <- ggplot(df_UCB1, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_UCB1$choice))),
           fill = as.numeric(sort(unique(df_UCB1$choice))))+
  labs(title="LinUCB (c=1.0)", x="Item id", y = 'Number of selections')+
  theme_minimal()
p1

p2 <- ggplot(df_TS, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_TS$choice))),
           fill = as.numeric(sort(unique(df_TS$choice))))+
  labs(title="TS", x="Item id", y = 'Number of selections')+
  theme_minimal()
p2

pall <- plot_grid(p2,p001,p01, p1)
pall


```

```r
#reinforcement
coins_bandit_contextfull <- OfflineReplayEvaluatorBandit$new(formula =  Loan_Status2 ~ loancat |Edu + Self_E + Male + Dependents +Married2 + Loan_Amount_Term + CoapplicantIncome + ApplicantIncome , data = loan_data_final, randomize = FALSE, replacement = FALSE,)
coins_bandit_context <- OfflineReplayEvaluatorBandit$new(formula =  Loan_Status2 ~ loancat | Male + Dependents + Married2 + ApplicantIncome  , data = loan_data_final, randomize = FALSE, replacement = FALSE,)
ts_bandit<- OfflineReplayEvaluatorBandit$new(formula = Loan_Status2 ~ loancat , data = loan_data_final, randomize = FALSE)


alpha=0.1
linUCB <- LinUCBDisjointPolicy$new(alpha=alpha)
linUCBfull <- LinUCBDisjointPolicy$new(alpha=alpha)
ts<- ThompsonSamplingPolicy$new(alpha = 1, beta = 1)

agent_lin <- Agent$new(linUCB, coins_bandit_context, name='Linear UCB')
agent_full<- Agent$new(linUCBfull, coins_bandit_contextfull, name='Linear UCB with complete features')
agent_TS <-Agent$new(ts, ts_bandit, name='TS')
# simulate
size_sim=100000
n_sim=5
simulator           <- Simulator$new(list(agent_full, agent_lin, agent_TS), # set our agents
                                     horizon= size_sim, # set the sizeof each simulation
                                     do_parallel = TRUE, # run in parallel for speed
                                     simulations = n_sim, # simulate it n_sim times,

)


# run the simulator object
history_coins           <- simulator$run()

# gather results
df_coins_result <- history_coins$data%>%
  select(t, sim, choice, reward, agent)


# check how many obs
max_obs_per_sim = df_coins_result%>%
  group_by(sim, agent) %>%
  summarise(max_t = max(t))
max_obs_per_sim

# Maximum number of observations
max_obs=1700

# data.frame aggregated for two versions: 20 and 40 arms
df_coins_agg_cumulative <- df_coins_result %>%
  group_by(agent, sim)%>% # group by number of arms, the sim
  mutate(cumulative_reward = cumsum(log(1 + reward)),
         rolling_reward = rollmean( reward, 252, na.pad=TRUE))%>%
  group_by(agent, t) %>% # group by number of arms, the t
  summarise(avg_cumulative_reward = mean(cumulative_reward),
            avg_rolling_reward = mean(rolling_reward),
            se_cumulative_reward = sd(cumulative_reward, na.rm=TRUE)/sqrt(n_sim),
            se_rolling_reward = sd(rolling_reward, na.rm=TRUE)/sqrt(n_sim)) %>%
  mutate(cumulative_reward_lower_CI =avg_cumulative_reward - 1.96*se_cumulative_reward,
         cumulative_reward_upper_CI =avg_cumulative_reward + 1.96*se_cumulative_reward,
         rolling_reward_lower_CI =avg_rolling_reward - 1.96*se_rolling_reward,
         rolling_reward_upper_CI =avg_rolling_reward + 1.96*se_rolling_reward)%>%
  filter(t <=max_obs)

```

```r


# create ggplot object
ggplot(data=df_coins_agg_cumulative, aes(x=t, y=avg_cumulative_reward, color =agent))+
  geom_line(size=1.5)+
  geom_ribbon(aes(ymin=cumulative_reward_lower_CI ,
                  ymax=cumulative_reward_upper_CI,
                  fill = agent,
  ),
  alpha=0.1)+
  labs(x = 'Time', y='Cumulative Reward of trading', color ='Algorithm', fill='Algorithm')
    +
  theme_bw()+
  theme(text = element_text(size=16))


# plot number of selections
# gather results
df_full <- df_coins_result %>%filter(agent=='Linear UCB with complete features')
df_lin <- df_coins_result %>%filter(agent=='Linear UCB')
df_thomp<- df_coins_result %>%filter(agent=='TS')

pfull <- ggplot(df_full, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_full$choice))),
           fill = as.numeric(sort(unique(df_full$choice))))+
  labs(title="LinUCB full", x="Item id", y = 'Number of selections')+
  theme_minimal()
pfull


pcontext <- ggplot(df_lin, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_lin$choice))),
           fill = as.numeric(sort(unique(df_lin$choice))))+
  labs(title="LinUCB rest", x="Item id", y = 'Number of selections')+
  theme_minimal()
pcontext

pTS<-  ggplot(df_thomp, aes(x=choice)) +
  geom_bar(color = as.numeric(sort(unique(df_thomp$choice))),
           fill = as.numeric(sort(unique(df_thomp$choice))))+
  labs(title="TS", x="Item id", y = 'Number of selections')+
  theme_minimal()
pTS

ptable <- plot_grid(pcontext,pfull, pTS)
ptable
```