barcelona gse
graduate school of economics

Master Thesis

Topic:

Unsupervised learning in decision making

Author:	Domagoj Fizulic
	Felix Gutmann
Student number:	125604
	125584
Program:	M.S. Data Science
E-Mail:	domagoj.fizulic@barcelonagse.eu
	felix.gutmann@barcelonagse.eu

I Table of Contents

I Table of Contents I

II List of Figures II

III List of Tables III

IV List of Listings IV

V List of mathematical symbols V

VI List of abbreviations VI

1 Introduction and conceptual approach 1

2 Relevant Literature 2

3 Theoretical Backround and experiments 3
- 3.1 Simmulation experiment design . . . 3
- 3.2 Reinforcement Learning backround and multi arm bandits . . . 4
- 3.3 Unsupervised Learning . . . 5
- 3.4 Simmulation results . . . 5

4 Data Analysis 5
- 4.1 Data Description . . . 5
- 4.2 Results . . . 5

5 Conclusion 5

6 List of Literature 6

Appendix A

II List of Figures

Fig. 1 Flowchart experiment desgin 3

III List of Tables

Tab. 1 Overview clustering algorithms 5
Tab. 2 Overview Outcome . . . . D

IV List of Listings

V List of mathematical symbols

Symbol	**Meaning**
a_t	Action at time t
$Q(a)_t$	Value function at time t
ϵ	Probability of exploration in epsilon greedy
α	Learning rate
τ	Softmax parameter
$H(X)$	Entropy of a discrete random variable X
$\mathbb{R}_0^+$	Positve real numbers including zero

VI List of abbreviations

Abbreviations	Description

1 Introduction and conceptual approach

Learning is a complex procedure. The learning procedure can affected due to social conditions...

2 Relevant Literature

3 Theoretical Backround and experiments

In this section we provide ...

3.1 Simmulation experiment design

Before analysing real data we run some simmulation experiments to indentify adequat approaches to model our data. To generate data artificially we follow a reinforcement learning and multi arm bandit approach where we let an agent learn the distribution of a simulated data set. The procedure is illustrated in figure 1. To keep things simple we draw a set of rewards from a normal distribution. We let several agents with different parameter setting process the reward data. From this procedure we obtain for each agent a time series of choices.

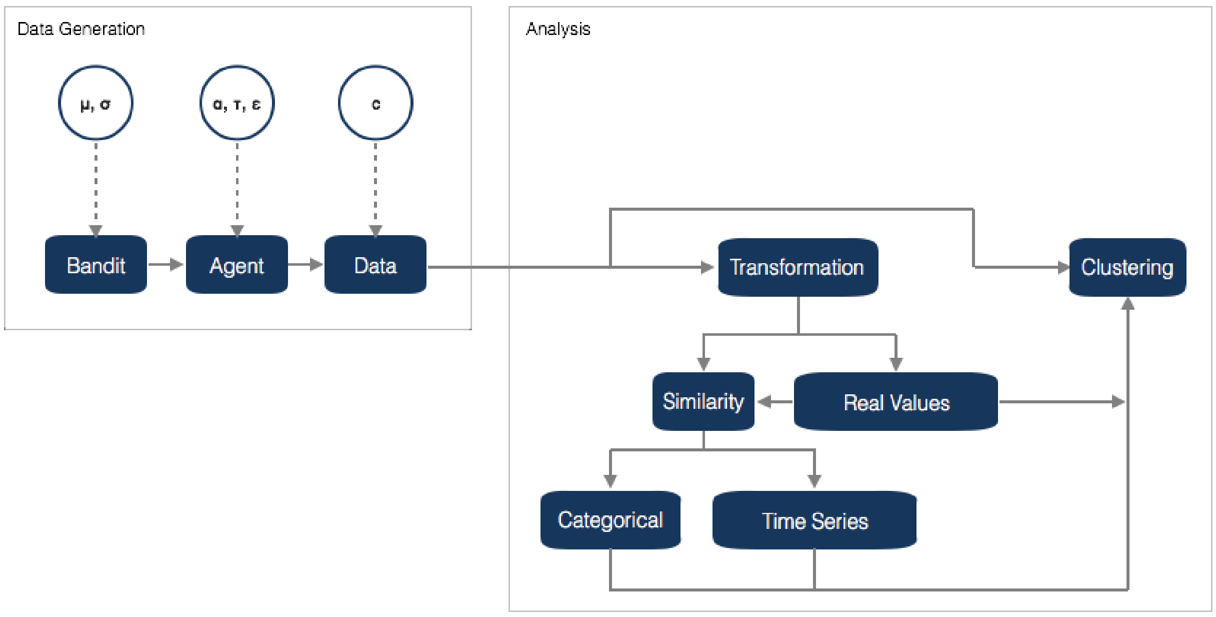

Figure 1: Flowchart experiment desgin

Concerning our data we find two main challanges. First, our data have a categorical nature. Furhtermore, the learning process also imposes a time series dependence on the data. We address this issue in the following ways. The first approach is to map the series of choices to a real valued series. For each time step we can compute *Shannon's Entropy* based on the empirical probability of the choices. Let X be a discrete random variable with probability p, then the entropy is defined as:

$$H(X) := -\sum_{i=1}^{N} p_i \log_2 p_i \qquad (1)$$

Entropy gives measure on how random a random variable behaves. Therefore, transforming choices to sequentiel entropy is to discriminate individuals by the randomness

3.2 Reinforcement Learning backround and multi arm bandits

Reinforcement Learning (RIL) is a branch of *Machine Learning* try model how an artificial agents interact with its environment and learns from the process over time.
In particular an agents is confronted with the task of choosing sequently from a set of choices. In comparison to *supervised learning*, where an agent is learning based on set of examples an agent in RIL doesn't have any knowledge about the system apriori. Therefore, it has to learn the nature of the system by sequentially interacting with its environment and keeping tack of the obtained information. Since the agent doesn't have any apriori information about the system it has to explore new possible action and so has to deviate from the optimal action. Furthermore, it has to keep track of value of each action he did so far. So the main task of the agent is to balance exploration and explotation. There are to basic approaches to model this trade-off; An *"Epsilon-Greedy"* selection method and Soft *"Softmax"* selection method. Before explaining both cocepts we introduce the value function for a given action a. Therefore, let $Q_t(a)$ be the value function of action defined as:

$$Q_t(a) := \frac{R_1 + R_2 + \cdots + R_{K_\alpha}}{K_\alpha} \tag{2}$$

The value function is the average over rewards.

Considering now epsilon greedy action selection mehtod: The rule in general is to select the next action as the current current highest value function. However, to model exploration we introduce a random element to deviate from that greedy strategy. Following that the next action

$$a_{t+1} = \begin{cases} \text{random action} & \text{, with probability } \epsilon \\ \arg\max_i Q_t(i) & \text{, with probability } 1-\epsilon \end{cases} \tag{3}$$

where $\epsilon \in [0,1]$ is a parameter controling the random behaviour of the agent.

In the softmax action selection method compute for each action a probability (also called *Boltzmann Distribution)*. The probability for action a is computed by:

$$P(a_t|X) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_i^K e^{\frac{Q_t(i)}{\tau}}} \tag{4}$$

In each iteration the next action of the agent is drawn with probability p_a:

$$a_{t+1} \sim p_a \tag{5}$$

After selecting an action the agent is updating its believe of the chosen action. Formally the update rule is defined

$$Q(a)_{k+1} = Q(a)_k + \alpha \left[R(a)_k - Q(a)_k\right], \tag{6}$$

where α is a is the non negativ *learning rate* defining how much the current action is affecting the believes.

3.3 Unsupervised Learning

We consider several unsupervised clustering techniques. A technical description for all of the is provided in

Algorithm	Input	Datatype
Spectral Clustering	Similarity Matrix	-
Affinity Propagation	Similarity Matrix	-
K-Means Clustering	Data Matrix	-
Ward Clustering	Data Matrix	-
PCA + Ward Clustering	Data Matrix	-

Table 1: Overview clustering algorithms

3.4 Simmulation results

4 Data Analysis

4.1 Data Description

4.2 Results

5 Conclusion

6 List of Literature

Appendix

Metrics and Similarities

This part of the appendix formally defines metrics and similarities and dissimilarities (proximity) used in this paper. We first define some basic general concepts followed by a description of the applied distance and similarity concepts.

Let $\boldsymbol{X}$ be a dataset and let $\boldsymbol{x_i}, \boldsymbol{x_j}$ be two datapoints, such that $\boldsymbol{x_i}, \boldsymbol{x_j} \in \boldsymbol{X}$.

A distance function assign for pairs a points a non negative real number as distance. $d : \boldsymbol{X} \times \boldsymbol{X} \mapsto \mathbb{R}_0^+$. Formally if the following properties are additionally staisfied the distance is also metric.

1. $d(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0$
2. $d(\boldsymbol{x_i}, \boldsymbol{x_i}) \geq 0$
3. $d(\boldsymbol{x_i}, \boldsymbol{x_j}) = d(\boldsymbol{x_j}, \boldsymbol{x_i})$
4. $d(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq d(\boldsymbol{x_i}, \boldsymbol{x_j}) + d(\boldsymbol{x_j}, \boldsymbol{x_i})$

A distance can be seen as a measure for disimilarity of two points. Besides distance some algorithms operate on a *similarity* matrix. Formally a similarity is a function $S : \boldsymbol{X} \times \boldsymbol{X} \mapsto [0, 1]$. Also for similarity we can define the following properties:

1. $0 \leq S(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq 1, \text{for } i \neq j$
2. $S(\boldsymbol{x_i}, \boldsymbol{x_j}) = 1$
3. $S(\boldsymbol{x_i}, \boldsymbol{x_j}) = S(\boldsymbol{x_j}, \boldsymbol{x_i})$

Once we have computed distance or a similarity we can compute for two data points we can use this information to transform it to a similarity or the distance vice versa:

$$S(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{1}{1 + d(\boldsymbol{x_i}, \boldsymbol{x_j})} \quad \Leftrightarrow \quad d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{1}{S(\boldsymbol{x_i}, \boldsymbol{x_j})} - 1 \tag{7}$$

Appendix

Principal Components and Multidimensional Scaling

B

Spectral Clustering

For the spectral clustering algorithm we formally introduce some graph notation. If not stated otherwise the following derivation follows [SOURCE]. In the following we consider a weighted and simple undirected graph.

$$G = \{V, E\} \tag{8}$$
$$V = \{v_1, \dots, v_n\} \tag{9}$$
$$E = \{e_1, \dots, e_n\} \tag{10}$$

Furthermore let the graph has a weighted and symetric ($|V| \times |V|$) adjacency matrix, such that:

$$\boldsymbol{W} = \begin{cases} w_{i,j} & \text{,if } v_i v_j \in E \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The *degree* of a node is defined as the sum of edge weights of connected nodes. Formmally we denote the degree of node i as:

$$d_i := \sum_{j=1}^{n} w_{ij} = \sum_{i=1}^{n} w_{ij} \tag{12}$$

Using the last expression we define matrix $\boldsymbol{D}$ as the diagonal matrix of the degress

$$\boldsymbol{D} := diag(\boldsymbol{d}) \tag{13}$$

The algorithm works on the *Laplacian* matrix defined by:

$$\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{W} \tag{14}$$

Former versions of the algorithm are applied on the graph laplcian. However, there were proposed newer versions using the so called *normalized laplacian*. Since also the python version is using this package we will focus on this version of the algorithhm. Following that the normalized graph laplacian is defined as:

$$\boldsymbol{L}_{norm} := \boldsymbol{D}^{1/2}\boldsymbol{L}\boldsymbol{D}^{1/2} = \boldsymbol{I} - \boldsymbol{D}^{1/2}\boldsymbol{W}\boldsymbol{D}^{1/2} \tag{15}$$

Specification	***μ***	***σ***	**CL Size**	**SD**	**Decision**	***α***	***τ***	**N**	**ALG**	**TRNS**	**MI**	**NMI**	**AMI**	**CS**	**HS**	**VMS**
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-		

Table 2: Overview Outcome