

# Master Thesis

Topic:

Unsupervised learning in decision making

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

# I Table of Contents

<b>I</b>	<b>Table of Contents</b>	<b>I</b>
<b>II</b>	<b>List of Figures</b>	<b>II</b>
<b>III</b>	<b>List of Tables</b>	<b>III</b>
<b>IV</b>	<b>List of Listings</b>	<b>IV</b>
<b>V</b>	<b>List of mathematical symbols</b>	<b>V</b>
<b>VI</b>	<b>List of abbreviations</b>	<b>VI</b>
<b>1</b>	<b>Introduction and conceptual approach</b>	<b>1</b>
<b>2</b>	<b>Relevant Literature</b>	<b>2</b>
<b>3</b>	<b>Theoretical Background and experiments</b>	<b>3</b>
3.1	Simulation experiment design . . . . .	3
3.2	Reinforcement Learning background and multi arm bandits . . . . .	3
3.3	Unsupervised Learning . . . . .	5
3.4	Simulation results . . . . .	5
<b>4</b>	<b>Data Analysis</b>	<b>5</b>
4.1	Data Description . . . . .	5
4.2	Results . . . . .	5
<b>5</b>	<b>Conclusion</b>	<b>5</b>
<b>6</b>	<b>List of Literature</b>	<b>6</b>
	<b>Appendix</b>	<b>A</b>

## 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
$\epsilon$	Probability of exploration in epsilon greedy
$\alpha$	Learning rate
$\tau$	Softmax parameter

## **VI List of abbreviations**

<b>Abbreviations</b>	<b>Description</b>
----------------------	--------------------

---

# **1 Introduction and conceptual approach**

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



## **2 Relevant Literature**

### 3 Theoretical Background and experiments

In this section we provide ...

#### 3.1 Simulation experiment design

Before analysing real data we run some simulation experiments to identify adequate 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 settings process the reward data. From this procedure we obtain for each agent a time series of choices.

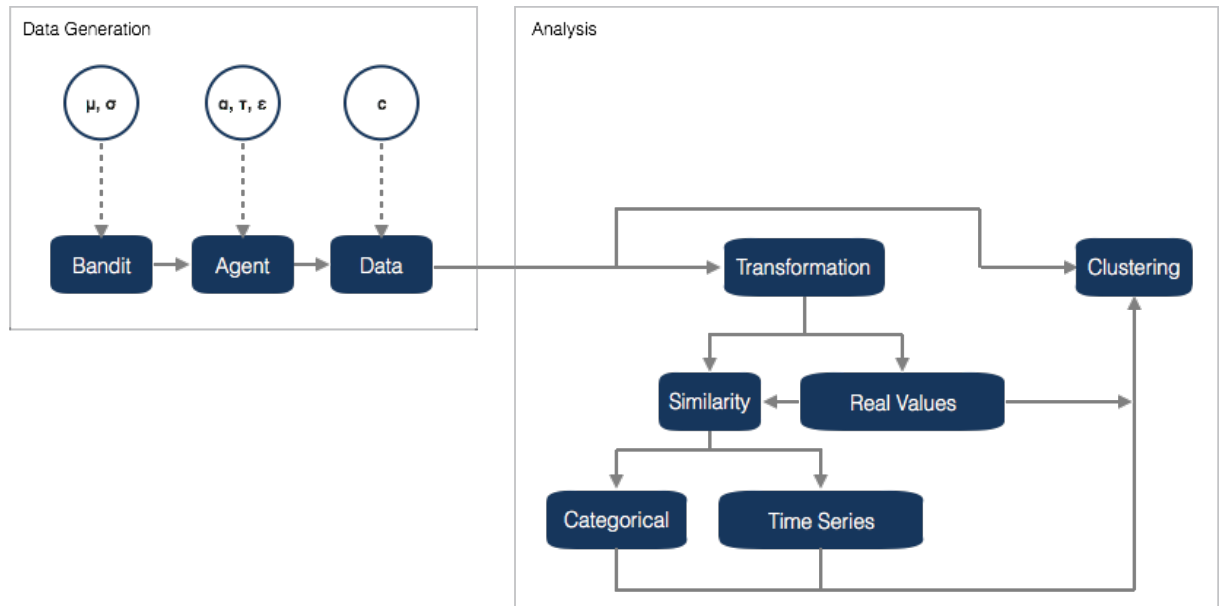


Figure 1: Flowchart experiment design

Concerning our data we find two main challenges. First, our data have a categorical nature. Furthermore, the learning process also imposes a time series dependence on the data. We address this issue in the following ways.

#### 3.2 Reinforcement Learning background 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 sequentially 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 track 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 exploitation. There are two basic approaches to model this trade-off; An "*Epsilon-Greedy*" selection method and Soft "*Softmax*" selection method. Before explaining both concepts we introduce the value function for a given action  $a$ . Therefore, let  $Q_t(a)$  be the value function of action defined as:

$$Q_t(a) := \frac{R_1 + R_2 + \dots + R_{K_\alpha}}{K_\alpha} \quad (1)$$

The value function is the average over rewards.

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

$$a_{t+1} = \begin{cases} \text{random choice} & , \text{ with probability } \epsilon \\ \arg \max_i Q_t(i) & , \text{ with probability } 1 - \epsilon \end{cases} \quad (2)$$

where  $\epsilon \in [0, 1]$  is a parameter controlling 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:

$$P(a_t|X) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_i^K e^{\frac{Q_t(i)}{\tau}}} \quad (3)$$

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

$$a_{t+1} \sim p_a \quad (4)$$

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

$$Q(a)_{k+1} = Q(a)_k + \alpha [R(a)_k - Q(a)_k], \quad (5)$$

where  $\alpha$  is the non negative *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	Data Matrix	-

**Table 1:** Overview clustering algorithms

### 3.4 Simulation 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  $\mathbf{X}$  be a dataset and let  $\mathbf{x}_i, \mathbf{x}_j$  be two datapoints, such that  $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ .

A distance function assign for pairs a points a non negative real number as distance.  $d : \mathbf{X} \times \mathbf{X} \mapsto \mathbb{R}_0^+$ . Formally there for axioms, which have to be staisfied:

1.  $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$
2.  $d(\mathbf{x}_i, \mathbf{x}_i) \geq 0$
3.  $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$
4.  $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$

Besides distance some algorithms operate on a *similarity* matrix. Formally a similarity is a function  $S : \mathbf{X} \times \mathbf{X} \mapsto [0, 1]$ . Also for similarity we can define the following properties:

1.  $0 \leq S(\mathbf{x}_i, \mathbf{x}_j) \leq 1$ , for  $i \neq j$
2.  $S(\mathbf{x}_i, \mathbf{x}_j) = 1$
3.  $S(\mathbf{x}_i, \mathbf{x}_j) = S(\mathbf{x}_j, \mathbf{x}_i)$

Once we have computed distance for two data points we can use this information to transform it to a similarity:

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + d(\mathbf{x}_i, \mathbf{x}_j)} \quad (6)$$

## **Principal Components and Multidimensional Scaling**

## 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.

$$G = \{V, E\} \quad (7)$$

$$V = \{v_1, \dots, v_n\} \quad (8)$$

$$E = \{e_1, \dots, e_n\} \quad (9)$$

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

$$\mathbf{W} = \begin{cases} w_{i,j} & , \text{if } v_i v_j \in E \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

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:

$$d_i := \sum_{j=1}^n w_{ij} = \sum_{i=1}^n w_{ij} \quad (11)$$

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

$$\mathbf{D} := \text{diag}(\mathbf{d}) \quad (12)$$

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

$$\mathbf{L} := \mathbf{D} - \mathbf{W} \quad (13)$$

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 algorithm. Following that the normalized graph laplacian is defined as:

$$\mathbf{L}_{norm} := \mathbf{D}^{1/2} \mathbf{L} \mathbf{D}^{1/2} = \mathbf{I} - \mathbf{D}^{1/2} \mathbf{W} \mathbf{D}^{1/2} \quad (14)$$



Specification	$\mu$	$\sigma$	CL Size	SD	Decision	$\alpha$	$\tau$	N	ALG	TRNS	MI	NMI	AMI	CS	HS	VMS
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

**Table 2:** Overview Outcome