DEMOCRATIC AND POPULAR ALGERIAN REPUBLIC

Ministry of Higher Education and Scientific Research

Blida 1 UNIVERSITY

Sciences Faculty

Computer Science Department

Master Project Thesis

Theme

« Reinforcement Learning Based Uncertain Pattern Mining. »

Presented By:

Mouloud Chaima

ElamraniFatimaZahra

Supervised By:

Mme.FatmaZohra Zahra

Examined By:

Mme.Yekhlef Hadjer

M.Riali Ishak

2021/2022

Thanks

First and foremost, we praise God Almighty for providing us with the strength and patience necessary to complete this work.
At the conclusion of our work, we would like to express our heartfelt gratitude to Mrs. ZAHRA for her invaluable advice and assistance during the duration of our project.

We'd also like to express our gratitude to all of our teachers who took part in our training throughout the course.

We thank our parents', families', and friends' encouragement and support.

Finally, we'd want to express our gratitude to everyone who has helped us construct this project.

Abstract

Pattern mining consists of finding interesting, useful and pertinent patterns (data structures) that exist among large amount of data. These discovered patterns can be used as actionable knowledge directly or they can be used by other data mining methods as an input. Itemsets represent the most basic type of pattern and are the most treated in this field. In the real world, the actual data is for the most part uncertain.

Indeed, we are interested in our work on this type of data, and as a result, our work consists of providing an approach for extracting frequent itemsets from uncertain data using deep reinforcement learning, which has had a lot of success in a variety of domains.

Keywords: frequent itemset mining, high utility itemset mining, uncertain data, reinforcement learning, deep learning.

Résumé

L'exploration de motifs consiste à trouver des motifs (structures de données) intéressants, utiles et pertinents qui existent parmi une grande quantité de données. Ces motifs découverts peuvent être utilisés directement en tant que connaissances exploitables ou peuvent être utilisés par d'autres méthodes d'exploration de données en tant qu'entrée. Les itemsets représentent le type de modèle le plus basique et sont les plus traités dans ce domaine. Dans le monde réel, les données réelles sont pour la plupart incertaines.

En effet, notre travail s'intéresse à ce type de données, et par conséquent, notre travail consiste à fournir une approche pour l'extraction d'items fréquents à partir de données incertaines en utilisant l'apprentissage par renforcement profond, qui a eu beaucoup de succès dans une variété de domaines.

Mots clés : extraction d'items fréquents, extraction d'items de grande utilité, données incertaines, apprentissage par renforcement, apprentissage profond.

ملخص

يتكون تعدين الأنماط من إيجاد أنماط مثيرة للاهتمام ومفيدة وذات صلة (هياكل البيانات) الموجودة بين كمية كبيرة من البيانات. يمكن استخدام هذه الأنماط المكتشفة كمعرفة قابلة للتنفيذ مباشرة أو يمكن استخدامها بواسطة طرق التنقيب عن البيانات الأخرى كمدخلات. تمثل مجموعات العناصر النوع الأساسي للنمط وهي الأكثر معاملة في هذا المجال. في العالم الحقيقي، البيانات الفعلية في معظمها غير مؤكدة.

في الواقع، نحن مهتمون بعملنا على هذا النوع من البيانات، ونتيجة لذلك، يتكون عملنا من توفير نهج لاستخراج مجموعات العناصر المتكررة من البيانات غير المؤكدة باستخدام التعلم المعزز العميق، والذي حقق نجاحًا كبيرًا في مجموعة متنوعة من المجالات.

الكلمات المفتاحية: التعدين المتكرر لمجموعة العناصر، التنقيب عن العناصر ذات المنفعة العالية، البيانات غير المؤكدة، التعلم المعزز، التعلم العميق.

TABLE OF CONTENTS:

General Introduction **01**

1. Context 01
2. Problem definition 01
3. Objective 02

Thesis Organization **02**

Chaptre1: Pattern Mining **03**

1.Introduction 04

2. Pattern Mining 04

3. Frequent Pattern Mining 05

3.1. Efficient Algorithms for mining frequent patterns 05

3.1.1 Apriori algorithm 05

3.1.2 FP growth algorithm 06

3.1.3 Eclat algorithm 07

3.2. Frequent itemset mining limitations 08

4. High utility pattern mining 08

4.1. HUP mining algorithms 09

4.1.1. Two Phase algorithms 09

4.1.2. One Phase algorithms 11

5. Uncertain data 13

5.1 Measures for calculating the frequencies of uncertain itemsets 13

i. Expected support 14

ii. Probabiliste Support 14

5.2 Algorithms for extracting uncertain frequent itemsets 15

5.2.1 U-Apriori 15

5.2.2 UF-Growth algorithm 16

5.2.3 HU-mine algorithm 16

5.3 Comparison between algorithms 17

5.4 Algorithms for extracting uncertain high-utility itemsets 18

5.4.1. PHUI _UA algorithm 18

6. Conclusion **19**

Chapter 2:Machine Learning 20

1. Introduction

2. Machine Learning 20

3. Applications of Machine learning 21

4. Machine learning model generation process description 22

5. Reinforcement Learning 22

5.1 Concept 23

5.2 Models 28

6. Deep Learning 28

6.1 Concept 30

6.2. Connexion Between Neurons in deep models 30

6.2.1. Feed Forward Network 31

6.2.2. Recurrent Neural Network 32

6.2.3. Deep Neural Network 33

7. Deep Reinforcement learning 34

8. Conclusion **36**

Chapter3: Proposed Approach 37

1. Introduction 37

2. Motivation Attributed to Using a Deep Model 37

3. Extraction of Uncertain Pattern Based on Reinforcement Learning (U-PMRL) 39

4. The architecture of the Deep Q-Learning Algorithm 41

5. Presentation of the Proposed Approach 45

6. Conclusion **46**

Chapter4: Testing and Experimentation 47

1. Introduction 47

2. Working Environment 47

2.1 Hardware Environment 47

2.2 Software Environment 49

3. Data Preprocessing 51

4. Model creation 55

5. Our Reinforcement Learning Model Performance 62

6. Conclusion **63**

General Conclusion and Perspective 63

1. General Conclusion 64

2. Perspective 64

- **Abbreviation List :**

FPM	Frequent Pattern Mining
ARM	Association Rule Mining
MinSup	Minimum Support
FP-growth	Frequent Pattern growth
FP-Tree	Frequent Pattern Tree
HUP	High Utility Pattern
UP-Growth	Uncertain Pattern growth
TWU	Transaction weighted utility
HTWUI	High Transaction Utilization Itemsets
HTWSP	High Transaction weighted Support Pattern
FHM	Fast High Utility
Minutil	Minimum Utility
AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
AGI	Artificial General Intelligence
ANI	Artificial Narrow Intelligence
ASI	Artificial Super Intelligence
NN	Neural Network
ANN	Artificial Neural Network
DNN	Deep Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
FFN	Feed Forward Neural Network
DQN	Deep Q Network
U-PMRL	Uncertain Pattern Mining Based on Reinforcement learning
ExpSupp	Expected Support

ProbSupp	Probabilistic Support

- **List of Tables**

Table1 The time needed to find all FI with probabilistic support measure 55

Table2 The time needed to find all FI (exp sup) with a different number of items 55

Table3 The time needed to find all FI (exp sup) with different number of transactions 56

Table4 The time needed to find all HUI with different number of transactions 56

Table5 The percentage of memory needed in extracting FI 57

Table6 The percentage of memory needed in extracting HUI 57

Table7 Comparison table between the result of the UApriori algorithm and our approach based on extracted FP from uncertain data 58

Table8 Comparison table between the result of the FHM algorithm and our approach based on extracted HUI from uncertain data 58

- **List of Figures**

Figure 01 Process of typical data analysis. [7] 04

Figure 02 FP_Tree. [2] 07

Figure 03 IHUP-Tree when min_util=50. [3] 11

Figure 04 The UH-struct for the probabilistic dataset of uncertain Data. [20] 17

Figure 05 A general RL workflow. [46] 22

Figure 06 Initial Q-Table. [47] 25

Figure 07 Q-Table after making an action. [47] 26

Figure 08 Updating Q-Table. [47] 26

Figure 09 Final Q-Table. [47] 27

Figure 10 The artificial and biological neuron. [15] 28

Figure 11 How a perceptron works. [15] 29

Figure 12 common activation functions. [38] 30

Figure 13 Total connection network. [39] 31

Figure 14 Partially connected networks. [39] 31

Figure 15 Recurrent Neural Network. [39] 31

Figure 16 The difference between Neural network and Deep neural network. [19] 32

Figure 17 Deep Reinforcement Learning [43] 33
Figure 18 The interaction cycle between the agent and the environment 39
Figure 19 Structure for Deep Q Learning 40
Figure 20 DQN architectures 41
Figure 21 How agent train to extract frequent itemsets 41
Figure 22 How agent train to extract high utility itemsets 43
Figure 23 Example of UBRecordlink data that we used 49
Figure 24 Encoding data step 50
Figure 25 The list of htwui from 29 items of UBRecordlink data 50
Figure 26 The list of htwsp from 11 items of UBSkin data 50
Figure 27 observation and action space of our environment with UBRecordlink data 51
Figure 28 Our DQN 52
Figure 29 Our state and actions for the ubrecordlink data 52
Figure 30 Our model state after training the agent on UBRecordlink data 52
Figure 31 Training step 54
Figure 32 The tuple Pattern of UBRecordlink data with min support = 7.905050446999999 with expected support measure 54
Figure 33 The tuple Pattern of the UBRecordlink data with min support = 7.905050446999999 with probabilistic support measure 54
Figure 34 The tuple Rule of the UBRecordink data with min_util=9188.1 55
Figure 35 Spmf software's interface 58
Figure36 Example of Extracted FI from the small data with spmf software 59
Figure37 Example of Extracted HUI from the UBRecordLink data with spmf software 60
Figure 38 The number of episodes needed to find all FI 60
Figure 39 The number of episodes needed to find all HUI 61
Figure 40 The cumulative reward as a function of the number of the first 10 episodes 61
Figure 41 The difference between the Q-Learning and the SARSA algorithm in the cumulative reward as a function of the number of the first 5 episodes in FI. 62

General Introduction

1. Context

The amount of data processed each day surpasses one billion, necessitating a very strong mathematical ability. The expansion in the size of data bases is what led to the development of information extraction techniques from data. [11]

Furthermore, enormous amounts of data are generated as a result of the quick development of applications in a number of real-world fields, including e-commerce and health. For instance, a business that saves consumer data for competitive reasons may be interested in adopting information extraction techniques. These methods often seek to identify patterns that frequently appear in the data that can be used to derive important data. However, the vast majority of these applications use uncertain data.

Exploration of motifs is a subset of data exploration that calls for finding interesting, unexpected, and useful themes within a set of data. Due to the numerous uses of uncertain data, the extraction of frequent sets of elements from uncertain data bases has gained significant attention in the data mining field.

Deep learning is a very fashionable field of artificial intelligence that has been explored in many fields in recent years, Deep learning is a type of machine learning that uses supervised and unsupervised algorithms to learn multi-level representations in hierarchical architectures, often used in classification and pattern recognition problems. One of the most interesting machine learning and artificial intelligence techniques is reinforcement learning, with numerous applications in a variety of disciplines

2. Problem definition

In the fields of data mining and machine learning, the problem of imperfect (missing, imprecise, inconsistent, and unclear) data, particularly uncertain data, is a well-known issue. The extraction of frequent itemsets from uncertain data has attracted the attention of scientists in the data mining field, and a large number of algorithms have been published in the literature as a result.

The scaling problem, on the other hand, is a common issue with these algorithms and their analogues that deal with accurate data.

3. Objective

The objective of our work is the creation of a new model that allows the extraction of frequent and high utility patterns from uncertain data using one of the most interesting machine learning techniques, Deep Reinforcement learning

Thesis organization

There are four chapters in the thesis:

Chapter 1: “Pattern Mining “

This chapter covers the many types of methods most typically used to extract frequent itemsets and high utility itemsets.

Chapter 2: “Machine Learning “

In this chapter, we'll go over the various machine learning techniques, with a focus on deep learning and the various architects.

Chapter 3: “Proposed Approach “

This chapter introduces the proposed approach to our problem as well as its architect.

Chapter 4: “Testing and Experimentation”

The final chapter illustrates and validates the method used.

Chapitre1: Pattern Mining

1. Introduction

We live in a world where large quantities of data are collected every day. Data analytics was classified as the first priority technologies, when actionable information can be extracted from the large volume of data available across the organization data analytics enables the company's stakeholders to make informed decision for their business, when business stakeholders make decision using data analytics capability this allows a company to have a greater likelihood of increasing revenue, reducing costs and improving its competitive advantage.[10]

This chapter covers the many types of algorithms most typically used to extract frequent itemsets and high utility itemsets.

2. Pattern Mining

Figure 1 shows the whole process typical of data analysis, among the various phases, data mining plays an important role consist of extracting from data stored in the database to make decisions and understand the data and action. Some of the most fundamental data mining tasks are pattern mining, so the main challenge in extracting data is to find useful information from massive amounts of data. Two important tasks of pattern mining are discussed in this chapter Frequent Pattern Mining and High Utility Pattern Mining. For example frequent pattern mining can be used in various applications over the world [11]. It can be used in product on shelves, wireless sensor networks, supermarkets for selling, and another applications that require user environmental monitoring.

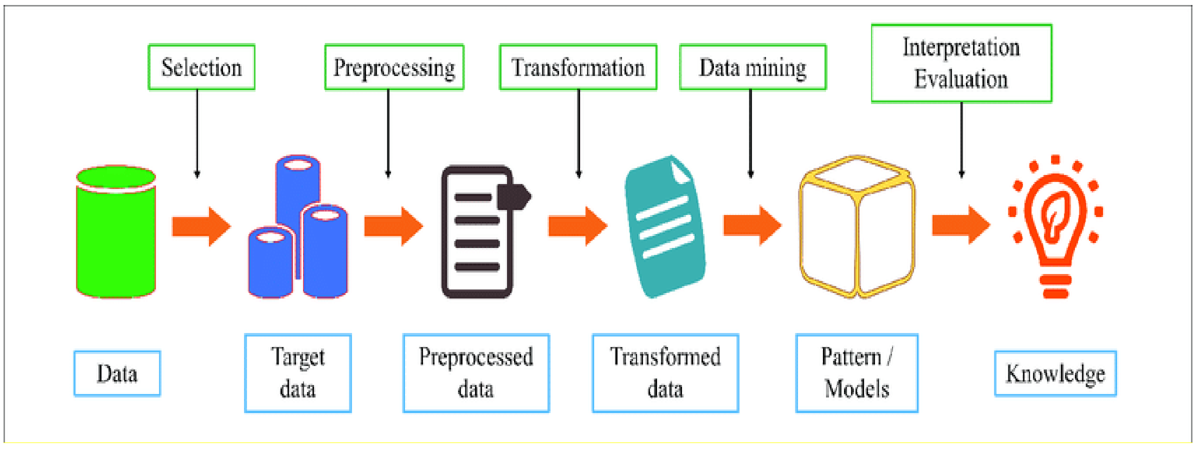


Figure 01 Process of typical data analysis [7].

Pattern mining consists of finding interesting, useful and pertinent patterns (datastructures) that exist among large amount of data. These discovered patterns can be used as actionable knowledge directly or they can be used by other data mining methods as an input.

3. Frequent Pattern Mining:

In data extraction research, frequent Pattern Mining (FPM) has been a major topic for many years and it plays an essential role in extracting rules. Considerable progress has taken place in this area and many effective algorithms have been designed to search frequent patterns in a transaction database.

In 1993,[1]Agrawal initially proposed this conception the form of association rule mining based on market analysis to determine association between items bought on the market, this concept used transactional Database for extract frequent patterns and interesting correlations, frequent patterns are subsequence or substructures of elements that appear in database transactions with a user-specified frequency, if a set of elements with a frequency is above or equal to the minimum threshold ,it will be considered a frequent pattern. Different techniques are applied to find frequent elements, there are two major problems relating to the technique of frequent Pattern mining, the first problem is that the database is parsed a few times, the second problem is the complex process of generating candidates with multiple sets of generated candidate articles, both problems are an efficiency patch infrequent pattern mining research shows that a great deal of effort has gone into this in order to innovate the best technology and many algorithms have proposed by different researchers to enhance the technology of FPM among them we mention : Apriori, FP_Growth, ECLAT,RARM and ASPMS algorithms[9]

3.1 Efficient Algorithms for mining frequent patterns:

3.1.1 Apriori algorithm:

Agrawal and Srikant [2] introduced this algorithm in 1994. Based on extracting frequent items, for generating association rules, Apriori uses an iterative approach in which k-itemset are used to explore (k+1) itemset and uses a breadth-first search to explore the search space of itemsets

Principle:

First, frequent items can be extracted by browsing the horizontal database to find frequent items L the resulting set is denoted by L_1 then using frequent 1-itemsets L_1 to generate items for two candidates L_2 and checking the database to get frequent items for two candidates , the set of frequent 2-itemsets used to find L_3 and so on. This process is repeated until no more frequent set of K_elements can be generated for a certain K, the finding of each $L_K$requires one complete database scan.

Second, the following principle underpins the Apriori property such as if a set of elements D does not satisfy the minimum support threshold then D is not frequent that is P(D) < sup-min, if an item A is added to the items D then the resulting item set (that is to say D ∪ A) cannot happen more frequently than D, hence D union A is not frequent no more that is P(D ∪A) < sup-min.

The final step is to generate Association Rules from Frequent Itemsets. After generalising the sets of frequent elements in transaction database D, strong association rules can be discovered, so that the strong association rules meet both the minimal support and minimum confidence, we explain this using the equation below.

$$\textbf{Confidence(A}\Rightarrow\textbf{B) =}\frac{(\text{support\_count}(A\cup B))}{(\text{support count}(A))}\ldots\ldots\ldots\ldots\ldots\ldots\textbf{(1)}$$

The number of transactions containing the (A∪B) items is represented by the support count(A∪B)and how many transactions included the A item is defined by support count (A).

Such that the association rules can be generated as follows:
1. Let X be the frequent set of elements, generate all the subsets of X.
2. For each non-empty subsets A of X, say that A⇒(X-A) if (support_count(X) /(spport_count(A)) ≥ min_conf, which represents the minimum confidence threshold.

3.1.2 FP_Growth Algorithm

Frequent Pattern Growth is an algorithm that extracts frequent sets without the costly candidate generation process.

As we have already said two main drawbacks of the algorithm Apriori :

- Firstly, the algorithm requires generating a large number of candidates.
- Secondly, it is time consuming to discover the itemset's support for each transaction in the database.

To treat these two problems of Apriori a new algorithm of frequent patterns called FPgrowth was developed by Han in 2000[3].

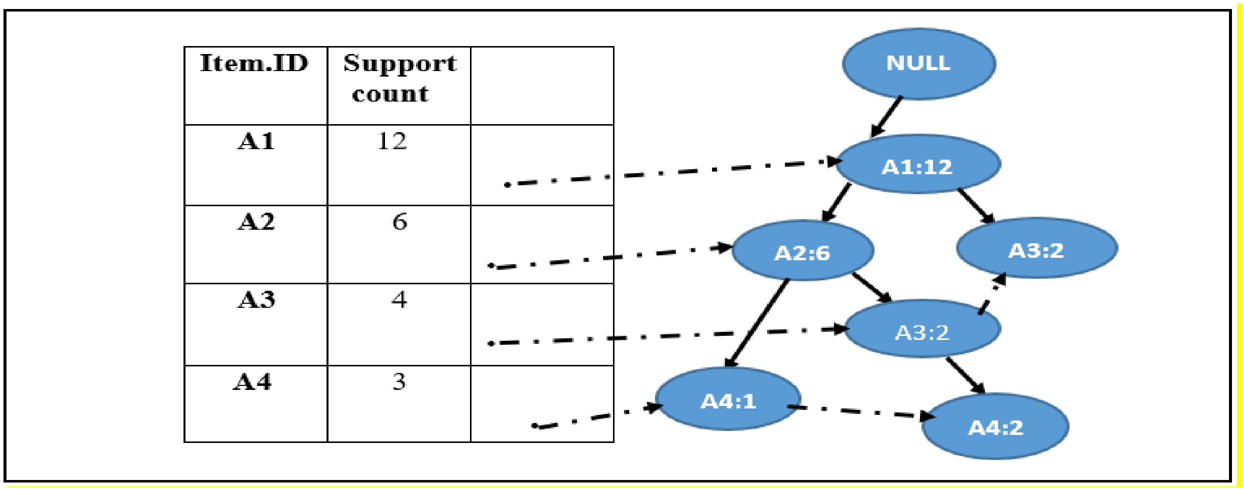
<u>**Principle:**</u>

The Algorithm based on two steps, the frequent pattern tree was created in the first stage by walking through the database twice. During the database's initial pass, it calculates the number of supports for each element and the data is analysed, such that the frequent patterns are ordered from the top to the bottom, and the infrequent patterns are removed from the list. On the second attempt of the database, build the FP tree (like it's shown in figure 02) then extract the frequent patterns from the FP tree using the FP Growth algorithm, it separates the collected database into a set of conditional databases, each linked to a frequent model and it exploits each database for each frequent model, consequently this method allows the size of the data sets to be searched to be reduced. The goals of the FP Growth Algorithm are to reach these important objectives:

-The first is that the database is scanned only twice, which will decrease the cost of calculation.

-The second key goal is to avoid generating a list of candidates.

-Finally, the algorithm reduces search space because it uses the divide approach.



Item.ID	Support count	
A1	12	
A2	6	
A3	4	
A4	3	

Figure 02 FP_Tree. [2]

3.1.3 Eclat Algorithm

In 2000, [8] Zaki proposed a new algorithm called Equivalence Class Clustering and Bottom-Up Lattice Traversal (ECLAT) which makes it possible to efficiently extract the sets of frequent items using the vertical data format. The Eclat algorithm which employs a Depth-First Search strategy, uses less memory than the Apriori.

Principle:

As already mentioned previously the Apriori and FP-Growth algorithms explore the frequent models of a transaction set in the horizontal form, but Eclat uses a vertical transactional representation and also in a transaction the elements are also considered to be ordered by lexicographical order.

By scanning the database once, from k=1 construct the candidates k+1 item sets frequent. Then, all itemsets with support less than min support will be eliminated.

Now, we will basically repeat the same thing as the previous step, but now for the pairs. The interesting thing about the ECLAT algorithm is that this step is done using the Intersection of the two original sets. This makes ECLAT faster because it is much simpler to identify the intersection of the set of transactions P_{id} than to scan each individual transaction for the presence of pairs (as Apriori does).

This process is repeated until all the frequent elements are generated and no set of frequent elements can be discovered. So, the Eclat algorithm produces frequent itemsets with their support as a result.

3.2. Frequent itemset mining limitations

When it comes to analyzing client transactions, frequent itemset mining has several limitations. Purchase amounts are not taken into consideration, which is a significant limitation. As a result, an item may appear only once or never in a transaction. So, it doesn't matter if a consumer buys five, ten, or twenty loaves of bread, it's seen as the same.

Second, the fact that all items are perceived as having the same relevance, utility, and weight is a second significant limitation. For example, if a consumer purchases a high-end bottle of wine or a simple loaf of bread, both are considered equally valuable.

As a result, frequent pattern mining may find a large number of uninteresting patterns. (Bread and milk) for example, could be a frequent pattern. However, this pattern may be unattractive from a commercial standpoint because it does not create much profit.

Furthermore, FPM algorithms that mine patterns often may miss rare patterns that have generated a large profit, such as caviar and wine.

4. High utility pattern mining

Due to its capacity to handle non binary frequency values of items in transactions and variable profit values for each item, high utility pattern mining has recently become one of the most significant research challenges in data mining.

When a database is updated or the minimum threshold is modified, incremental and interactive data mining allows you to reuse previous data structures and mining results to eliminate not necessary calculations.

Many research have concentrated on classic frequent pattern mining, which only considers the presence of patterns in the database and ignores the internal utility values (such as quantity) and external utility values (such as value, profit, and price) of each item in the itemset[12].

To address this problem, high utility itemsets (HUIs) mining has been used in a variety of fields, such as website clickstream analysis [13, 14], mobile commerce environment [18], cross marketing commercial value of retail stores [19], user behaviour analysis, Web mining, bioinformatics, market basket analysis, and promoting HUIs sales to increase profit.

4.1. HUP mining algorithms

Many existing HUPs mining techniques may be divided into two types: two-phase and one-phase approach [14].

4.1.1. Two phase algorithms

The algorithm requires two phases to identify all HUPs:

i. The candidate itemsets are generated in the first step by finding the utility value of each candidate itemset.

ii. The real utility value of each candidate itemset in the second step is calculated by the scan of the dataset.

The algorithms TwoPhase [13], IHUP [14], UP-Growth [15], HUP-Growth [16], and MUGrowth [17] use this two-phase technique.

- **Two Phase algorithm :**

The Apriori method is generalised by the Two-Phase algorithm [14]. The main distinction between Apriori and Two-Phase is that the two phase includes a second phase in which the utility of each pattern calculated with database scanning.

The result of Two-Phase is a list of high utility itemsets with a utility greater than the user-defined minimum utility threshold. On the other hand, Two-Phase has significant performance limits. The first is that, because Two-Phase creates itemsets by joining itemsets without consulting the database, it can produce patterns that aren't even in the database. As a result, Two-Phase can waste a lot of time processing itemsets that aren't in the database.

The second disadvantage is that Two-Phase scans the database many times to compute the TWU and utilities of itemsets, which is very expensive.

Third restriction is that utilizing a breadth-first search might be memory expensive because it needs keeping all k-itemsets and (k-1)-itemsets in memory at all times (for $k > 1$).

• IHUP Algorithm

Ahmed et al [5] presented IHUP, a tree-based approach for extracting high utility itemsets.

This pattern-growth algorithm's basic principle is to search a database for itemsets and prevent generating itemsets that don't exist in the database. Moreover, it has created compact database representations and the notion of projected database to lower the size of databases when an algorithm explores bigger itemsets to reduce the cost of scanning the database.

It employs the IHUP tree to keep information of high utility itemsets and transactions. Each node of the IHUP tree has the item name, medium, and TWU value.

So, this algorithm include into three steps:

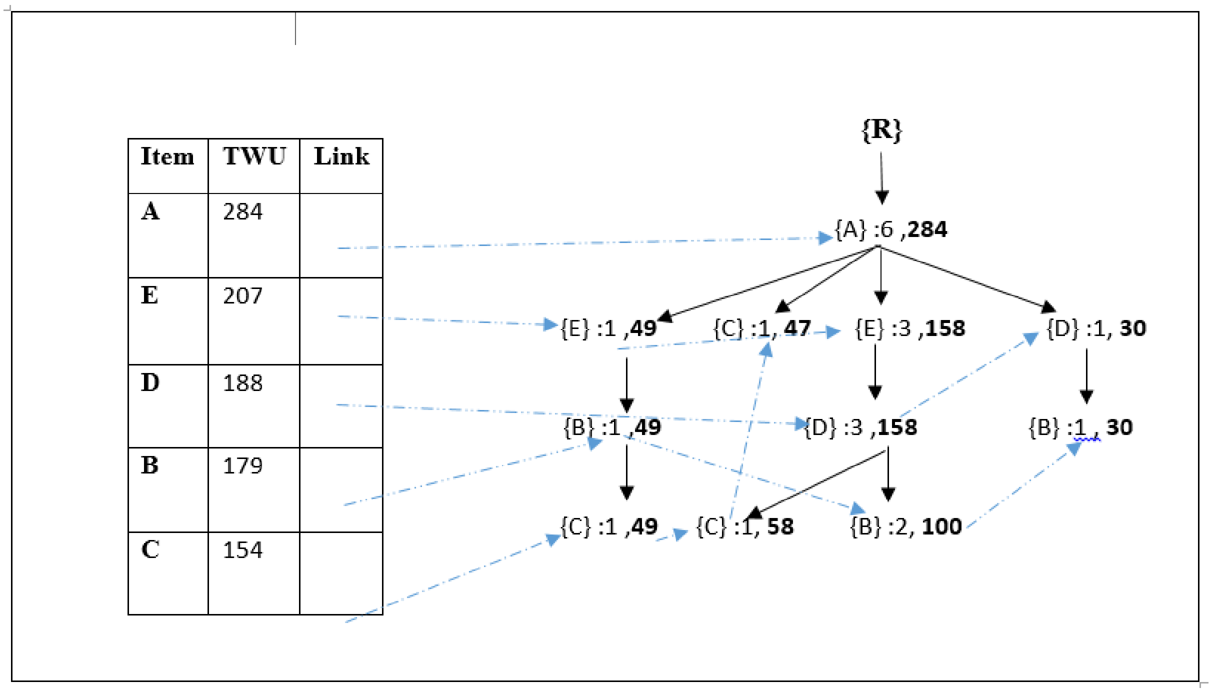
1-Exploration of the search space of itemsets using a depth-first search rather than a breadth-first search to build the IHUP tree structure, the advantage of using that throughout the search is that fewer itemsets will be stored in memory.

2-The creation of HTWUI (if TWU(x) is more than the minimal utility threshold, x is referred to as a high transaction weighted utilization Itemset).

3-To identify a hui: First step, the elements of the transaction are not ordered in a predefined sequence (order reduction support or TWU) The rearranged transactions are subsequently entered into the IHUP tree, as shown in the image, which illustrates the table database IHUP tree in decreasing order of TWU elements.

Second step, the FP-Growth method is used to build HTWUI from the IHUP tree.

Third step, by evaluating the original database once, a collection of high utility items and their utility is determined from the HTWUI collection.



Item	TWU	Link
A	284	
E	207	
D	188	
B	179	
C	154	

Figure 03 IHUP-Tree when min_util=50.[3]

4.1.2. One phase algorithms

The invention of methods that do not create candidates was the second important advance in high utility itemset mining. These one-phase methods calculate the utility of each pattern in the search space. As a result, an itemset may be quickly classified as either low or high utility, and patterns do not need to be stored in memory.

The one-phase algorithm concept was initially introduced in HUI-Miner and later in the d2HUP [19] method. The speed of d2HUP and HUI-Miner is significantly faster than that of two-phase algorithms.

FHM, mHUIMiner, ULB-Miner, HUI-Miner*, and EFIM are examples of enhanced and more efficient one-phase algorithms.

- **The FHM (Fast High-Utility Miner) Algorithm :**

FHM is a one-phase method that explores the search space of itemsets by doing a depth-first search. The FHM algorithm, for each visited itemset in the search space creates a utility list throughout the search. An itemset's utility-list stores information about the itemset's utility in transactions where it occurs, as well as information about the utilities of remaining items in these transactions. Utility-lists structure allows to directly obtain the utility of an itemset, and the upper-bounds on the utility of its supersets without scanning the database. Furthermore, by linking utility-lists of shorter patterns, utility-lists of k-itemsets k > 1 may be efficiently generated.

The main procedure of FHM is to:

First, take the quantitative transaction database as input and the minutil threshold. FHM initially scans database D to determine each item's TWU (transaction-weighted utility), which is calculated using the equation:

$$\text{TWU(X)}=\sum_{T_{id}\in D \wedge x \sqsubseteq T_{id}} Tu(T_{id}) \text{ ……………….} \mathbf{(2)}$$

Second, get the set of all items with a TWU of at least minutil. The TWU values of the items are then utilized to create a total order on the items, which is the ascending TWU value order.

The database is then scanned, items in transactions are reordered according to the total order during this database scan, the utility-list of each item is generated [15], and a structure called EUCS (Estimated Utility CoOccurrence Structure) is built [16].

Following the building of the EUCS, the recursive FHM Search is used to begin the depth-first search examination of itemsets. This search technique works like this: For each extension $P_{id}$of an itemset, if the total of the iutil values of P_{id}'s utility-list is no smaller than minutil, Pid is a high-utility itemset, and it is output.

After that, If the sum of iutil (the utility of X in Ttid: u(X, Ttid) and rutil (defined as: $\sum_{i\in T_{id} \wedge i>x \wedge \forall x\epsilon X} u(i, T_{id})$) values in P_{id}'s utility-list are greater than minutil, indicating that P_{id}'s extensions must be explored. This is accomplished by combining $P_{id}$with all ofP_{id}'s extensions.

To merge the utility-lists of itemset, $P_{id}$and the merging P_{id} with all extensions of P_{id}, the utility-list of the merging P_{id} with all extensions of $P_{id}$is created. Then figure out how useful it is and how far it may be developed (s).

Because of the F H M Search method starts with a single item, it recursively searches the search space of itemsets by adding single items and only prunes the search space depending on, If TWU (X) ≥ min_util, then the itemset X is HUIs.

The FHM algorithm's utility-list structure is described as a vertical database representation.

- Utility-list algorithms have been demonstrated to be more than two orders of magnitude quicker than two-phase algorithms [12,13,18]. On the other hand, Utility-list-based algorithms had significant limitations.

  First, because itemsets are created by combining itemsets rather than reading the database, these algorithms may explore certain itemsets that never occur in the database. As a result, these algorithms may waste a significant amount of time generating utility-lists for itemsets that do not exist.

  Second, because a utility-list must be generated for each visited itemset in the search space, these methods can use a lot of memory. An itemset's utility list can be extremely long. In the worst-case situation, it contains a tuple for each database transaction.

- Because two or three utility-lists must be compared to create the utility-list of each k-itemset (k > 1), the join operation can be very expensive, the ULBMiner method was recently introduced by expanding HUI-Miner and FHM to lower the memory demand of utility-list based algorithms. ULB-Miner uses a buffer to reuse memory for utility-list storage. This method has been demonstrated to enhance both runtime and memory use, HUI-Miner*, which depends on an updated utility-list structure to speed up HUI-Miner, is another upgrade.

5. Uncertain data

In the exploration of frequent patterns an important aspect to consider into account is whether the data to be used is certain or uncertain, in terms of precise data (certain) no probability is assigned to the data , However with uncertain data, each element of the database has a probability value between 0.0 and 1.0 such that the concept of frequent exploration of models from probabilistic databases because the data generated by many applications is uncertain, it has become extremely important. For example:

- Data is updated in a privacy-preserving application by modifying its state. The format of the output [21] in this sort of application is the same to that of the uncertain data. [20]

5.1 Measures for calculating the frequencies of uncertain itemsets

Pattern mining algorithms designed for exact data can't be used on imprecise data, such that the frequency of occurrence is often expressed in terms of the number of supports, however for datasets containing uncertain elements the definition of support must be redefined what encourage scientists to create new algorithms for extracting frequent itemsets from uncertain data, these algorithms use two measures to calculate frequent itemsets:

i. Expected support:

The total of the existential probability P(X,t_j) of X in transaction t_j across all n transactions in the probabilistic database is the expected support **expSup(X)** of pattern X in the whole database.

The Expected support function expression and present as follows:

$$\text{expSup}(X) = \sum_{j=1}^{n} P(X, t_j) = \sum_{j=1}^{n} (\prod_{x \in X} P(x, t_j)) \text{..................}\textbf{(3)}$$

When elements x ∈ X in each transaction t_j are independent.

Where j represents the number of transactions.

If and only if **expSup(X)≥minsup** , a pattern X is considered to be a frequent pattern in a probabilistic database.

ii. Probabiliste Support :

An itemset X is a probabilistic frequency if its existence in a minsup transaction is higher than or equal to the probabilistic minimum threshold set by the specific user (MinProb) [22].

$$P(\text{Sup}(X) \geq \text{MinSup}) \geq \text{MinProb} \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\textbf{(4)}$$

In the database of uncertain transactions the support of one or more elements must be characterized by a discrete probability distribution rather than a single value [22].

Let T be the transaction database and W the set of possible elements of T, the $P_i(X)$is the support probability of an itemset X such that X has support i.

$$P(X) = \sum_{w_j \in W, (s(X, w_j) = j)} P(W_j) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\textbf{(5)}$$

Where s(X,W_j) denotes X's support in an elementW_j.

Let I be the set of all possible elements of T represents the uncertain database where a transaction $t_j \in T$ is a set of uncertain elements namely $t_j \sqsubseteq I$, contrary to the certain database each item $x_i \epsilon t_j$ is associated with an existential probability P(x_i, t_j) ∈ [0,1] which denotes the probability that xi is present in t_j, for an itemset X⊑ t_j based on the assumption that the elements of X are independent the existential probability

P(X⊑ t_j)=$\prod_{x \in X} P(X \sqsubseteq t_j)$,Expected support of X is then the sum of the probability over all transactions[22] **,**the frequent probability P(X) of X calculated by summing Pi(X) for all

I≥MinSup[20].

$$P(X)=\sum_{i=minsup}^{|T|} P_i(X) \text{...} \textbf{(6)}$$

Where Pi(X) is the probability of exactly X occurring in one transaction:

$$P_i(X)=\sum_{\substack{S\sqsubseteq T\\ |S|=i}} \left(\prod_{t\in S} P(X \sqsubseteq t) \prod_{t\epsilon T-S} 1-P(X \sqsubseteq t)\right. \text{ ....} \textbf{(7)}$$

So, if P(X)≥MinProb, X is a probabilistic frequent itemset.

5.2 Algorithms for extracting uncertain frequent itemsets

For extracting important information with uncertain data ,There have been a variety of algorithms developed for researching frequent models. While processing uncertain data, we note the algorithms UF-growth, U-Apriori, UFP-growth, PUF-growth, UH-min, and TPC-growth. So we have identified a certain algorithm with the calculated measures seen previously for the extraction of frequent itemsets from uncertain data.

5.2.1. U-Apriori algorithm

It is an extension of the Apriori algorithm introduced by [20]Chui et al.to deal with uncertain data.

The only modification is that in the Apriori algorithm the number of support of the candidate model increased as a result of their true support on the other hand in the U-Apriori algorithm the support of a given model is incremented by the expected support, The approach is based on the downward closure property, which means that every non-empty subset of a set of frequent items must also be frequent.

The steps are explained as follows:

Step_1: The algorithm examines the uncertain database to get the expected support S of each 1-itemset, then compares the expected support to the minimal support to determine the frequent 1-itemset L1.

Step_2: The algorithm generates K-itemset(2-itemset) using L_{K-1} (1-itemset) and prunes the infrequent itemset with the APriori property.

Step _3: Re-parse the database to obtain the k-item candidate set's support, compare it to the minimal support, and obtain the frequent k-itemset L_K.

Step _4: generate all non-empty subsets of each frequent element set.

Step _5: for any non-empty part output the rule "S(1-S)".

The two main disadvantages of this algorithm are that the number of candidates to generate takes more memory space and a large exclusion time and another problem is the larger number of database analysis for frequent model generation. When the database is huge, the algorithm does not work effectively.

5.2.2. UF-Growth algorithm

Leung et al[20] proposed the UF-Growth algorithm. to effectively represent uncertain data ,FP-Growth is a variation of this algorithm. The UF-Growth algorithm functions as follows:

Step -1: To determine the expected support, the algorithm first examines the database. of each element to find the frequent elements having an expected support S greater than the minimum support.

Step -2: It orders the frequently occurring items in order of expected support in declining order.

Step -3: The algorithm analyzes the database once more and inserts each transaction in the UF tree as in the FP tree, with the exception that each UF-tree node contains the element name X, its expected support, and its number of occurrences, which indicates how many times such expected support exists for an element in the database. The difference is that the path in the UF tree is merged only if the tree nodes on the path have the same element name and existential probability.

The UF growth algorithm identifies frequent patterns from the UF tree structure in the same way as the FP growth algorithm does, with the exception that:

- We must preserve the expected support of the X instead of the actual support when the projected database for a model X is trained in an uncertain database.
- We estimated the expected support of y and x, for example X{y}, while calculating the expected support of an extension of the model X.

5.2.3. UH-mine algorithm

Proposed by Agrawal [20] is a variant of the H-Mine[23] algorithm , the algorithm uses an array structure known as UH-struct which stores the probability of each element and the link for element the algorithm works as follows:

Step -1: According to the expected support, the UH-mine algorithm scans the database to find all the frequent elements.

Step -2: The algorithm builds a main table which stores all the frequent elements with their expected support for each element the header table contains the name of the element, the expected support and a size.

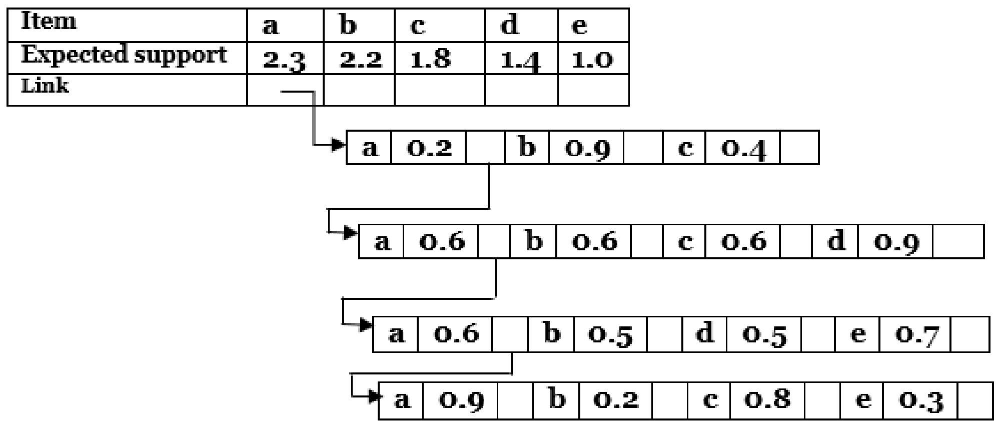


Figure 04 The UH-struct for the probabilistic dataset with uncertain Data.[20]

Step-3: Insert transactions into the UH-structure data structure after building the header table.
Step-4: The algorithm builds the header tables repeatedly where distinct item sets are prefixed and produces the frequent item sets.

5.3 Comparison between algorithms

In order to evaluate the benefits and drawbacks of using each algorithm's efficacy in the field of data mining, this section examines the frequent pattern mining methods for uncertain data that were mentioned in the previous section. We will contrast the previously mentioned algorithms in the table below.

Algorithms	**Advantage**	**Disadvantage**
U-Apriori	-provide precise results[41] -The candidate set size is greatly reduced by the Apriori property [1].	The performance is impacted by the repeated database scans.
UF-growth	-By searching the database twice, this technique typically exploits uncertain database models.	-Greater tree size than the FP-tree [21]
UH-mine	-superior over the three earlier algorithms [41].	-Memory use are increased using UH-Struct [41]

5.4 Algorithms for extracting uncertain high-utility itemsets

Based on the notion that uncertainty is frequent in real-world applications, researchers are proposing algorithms to efficiently extract high-utility items from uncertain databases such as : PHUI-UP algorithm ,PHUI-list .

A new PHUIM framework, called Potentially High-Utility Itemset Mining [23], in uncertain databases, is proposed to effectively identify not only high-utility but also high-probability elements. The probability measure that is similar to the expected support based frequent item extraction model is used in the proposed PHUIM framework, and we dubbed it the potential probability measure. The uncertainty model utilized in PHUIM is quite similar to the model used for probabilistic databases.

5.4.1. PHUI-UP algorithm

The PHUI-UP algorithm is designed to extract PHUI from uncertain datasets using a level approach based on an Apriori type approach where a dataset's itemset X is considered a potential high-utility itemset (PHUI) if it meets both of the following criteria:

1-X is a HUI , An itemset is considered to be a high-utility itemset (HUI) in a database if its utility value u(X) is more than the minimum utility count as:

$$\sum_{X\subseteq T_q \cap T_q \in \mathrm{D}} U(X\,,T_q) \;\geq Tu\; \times \varepsilon \text{ ....}\textbf{(8)}$$

2- X is a HPI , If the potential probability of an item set X is more than or equal to the minimal potential probability , which is determined as follows:

$$\sum_{X\subseteq T_q \wedge T_q \in \mathrm{D}} P(X\,,T_q) = Pro(X) \geq \mu \times |D| \text{ .... } \textbf{(9)}$$

The algorithm works in two phases as following:

Step-1: The algorithm scans the database in the first phase to determine TWU, TU values, and probabilities for all 1-itemsets Pro(X) in the database, which are defined as:

$$tu(T_q)=\sum_{j=1}^{m} U(i,T_q) \text{ ....}\textbf{(10)}$$

$$Tu(X) = \sum_{T_q \in D} tu(T_q)\text{..................................}\textbf{(11)}$$

$$Pro(X) = \sum_{T_q \in D} P(X,T_q)\text{.............................. }\textbf{(12)}$$

Then, the algorithm computes HTWPUI which is defined as, If TWU(X)≥TU×ε , an itemset X in D is defined as a high transaction-weighted utilization itemset (HTWUI).

Step-2: The HTWPUIs are then constructed based on the designed candidates of $\mathrm{HTWPUI}_{\mathrm{k}}$ (where k is initially set to 1), and will then be utilized to generate the following candidates C_{k+1}discover $\mathrm{HTWPUI}_{\mathrm{k+1}}$.

Step-3: the database must be rescanned to find $\mathrm{HTWPUI}_{\mathrm{k+1}}$ in each level C_{k+1}, phase one ends when there isn't a candidate created

Step-4: The algorithm does the second phase in such a way that a database analysis is essential to discover the final PHUI from HTWPUI.

6. Conclusion

We have studied in this chapter various algorithms of pattern mining from either precise or uncertain data. We can distinguish between these algorithms by different metrics such as, the data structures, techniques they employ to scan the search space, they examine in depth first or in broad first, internally or externally, the kind of database representation they utilize (vertical /horizontal), how they come up with or decide on the next itemsets to examine in the search space, how they decide if itemsets satisfy the minimal utility constraint by calculating their utility, What type of data structure is utilized to store transaction and item information (e.g. hyperlink structures, utility-lists, tree-based structures), whether the data is certain or uncertain.

Chaptre2: Machine Learning

1. Introduction

Artificial Intelligence can be divided into three categories, which are: Artificial narrow intelligence (ANI) which is known as machine learning; these systems are designed to handle a particular problem and do a single task very effectively; it comes in the first stage. In the second stage, Artificial General Intelligence (AGI) which is known as Machine Intelligence and a machine that is as smart as a person. And in the third stage, Artificial Super Intelligence (ASI) will be able to transcend all human skills, known as Machine Consciousness.

Machine Learning (ML) provides us with the mathematical skills we need to duplicate and mimic human behaviour, which is the objective of AI. With the use of machine learning algorithms, AI can understand language and carry on a conversation, allowing it to continuously learn and better itself based on experience. So, like humans, ML learns from data in order to execute a higher-level function.

In this chapter, the important methods and models of ML where presented and discussed as well as its types.

2. Machine Learning

Machine learning is a subset of AI that focuses on the creation of computer programs that have access to this current data by allowing the system to learn and improve automatically by detecting patterns in the database without the need for human interaction. And without being explicitly programmed to do so, as defined by Arthur Samuel and Tom Mitchell [23].

3. Applications of Machine learning

It has already infiltrated every aspect of our life without our knowledge. Almost every machine we use, as well as the high technology machines we have seen in the previous decade, has incorporated machine learning to improve product quality. The following are some cases of machine learning:
Image recognition used in social media by Facebook to help tag people in posts, it is also used in most phone camera apps to recognize objects in images. Another application of ML is speech recognition used by voice assistants like google assistant for android and Siri for apple. Product recommendation based on user preferences in shopping websites, streaming apps ...etc. Email spam filtering used in most mailing services such as Gmail, outlook ...etc. Automatic language translation which can be combined with image recognition to translate even text in pictures.

4. Machine learning model process

- **Data identification:**

This is the initial step, identifying the various data sources that will be obtained and merging them as needed. The quality of data used is essential since it has a direct impact on the model's output.

- **Data preparation:**

After the data has been collected and obtained as a dataset, the Data Preparation stage begins, in which the characteristics of the data are recognized, as well as the format and quality of the data. All of the data is collected into a dataset and randomized, so the majority of it can be used to train the model and the remainder may be used to test it and to validate.

- **Data cleaning:**

In this step, data is transformed into the proper format, and feature identification and labeling are accomplished during this stage. The dataset is cleaned up by removing data with missing values, duplicate values, and invalid data. This assures that the machine learning algorithm is not affected by noise in the data.

- **Data analysis:**

In this stage, we will start slicing and dicing our data in order to extract useful information. We will seek for hidden patterns and correlations, as well as insights and forecasts, using various data visualization tools.

- **Training:**

The dataset created in the preceding phases is utilized to train the algorithm and construct the model in this step. Using characteristics and labels provided in the dataset, the algorithm detects different patterns in the data and predicts the outcome for a particular input.

- **Testing:**

This is one of the most crucial processes in determining if the trained model produces the desired result. The data from the dataset is utilized to verify the model's result.

- **Deployment:**

We go on to the last stage of deploying the model after getting the desired accuracy.
So, the trained model is then deployed and brought to life in the real-world system.

5. Reinforcement Learning

5.1 Concept

There are several ways to classify the different types of Machine Learning Algorithms, however they can typically be divided into classes based on their motivation, so reinforcement learning is one of them.

Reinforcing learning is the concept of learning by doing, which is based on the psychology concept of reinforcement behavior. By trial and error, a machine can arrive at an optimum result. It learns to pick particular activities that lead to the desired outcome over time.

Reinforcement learning differs from supervised learning in that supervised learning includes the answer key, allowing the model to be trained with the correct answer, whereas reinforcement learning does not include an answer and instead relies on the reinforcement agent to decide what to do to complete the task. It is obligated to learn from its experience in the absence of a training dataset[25].

This type of machine learning is generally just concerned with increasing the function's effectivness.

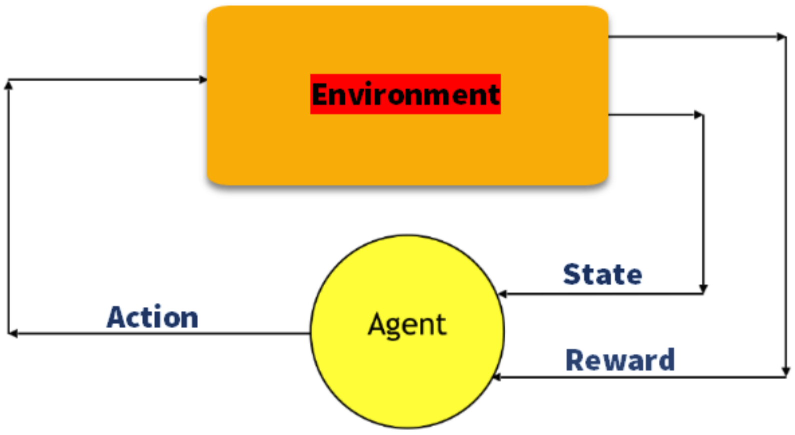


Figure 05 A general RL workflow. [25]

Like it's shown in this figure, Within the RL workflow, five different areas need to be addressed:

Agent: An RL agent is a program that we're learning to make good choices (for example: a Robot that is being trained to move around a house without crashing).

Environment: The agent's environment is the space in which he or she interacts (the house where the Robot moves). The agent has no control over the environment; all it has is control over its own activities (the Robot cannot control where a table is in the house, but it can walk around it to get out of crashing case).

State: The agent's current situation is defined by the state (it could be the exact position of the Robot in the house, or the alignment of its two legs, or its current posture; it depends on how we address the problem).

Action: The decision made by the agent in the current time step (moving its right or left leg, or raise its arm, or lift an object, turn right or left, etc.). We know ahead of time what actions (decisions) the agent can take.

Policy: A policy is the intellectual process that goes into deciding an action. It is a probability distribution attributed to the collection of actions in practice. Actions that are highly rewarding will have a strong probability, and vice versa. It's important to note that just because an action has a low probability doesn't mean it won't be chosen. It's only that it has a lower chance of being chosen.[26]

5.2 Models

There are two important learning models in reinforcement learning:

- **Q learning**

The best known and most widely used is algorithm Q-learning which is a value based and an out-of-policy reinforcement learning algorithm, such that Quality is represented by the "Q" in Q-learning. The latter represents the usefulness of a given action to obtain future rewards.

Because the q-learning function learns actions that are not covered by the present policy, such as random acts, it is deemed off policy. Specifically, the goal of q-learning is to determine the optimum action in the current situation and get maximum rewards [24].

➢ The principle of Q-learning consists of:

- Build the action-value function Q (knowledge of the agent) this function is summarized by an association of each state-action couple (s_j, a_j) with an estimated value V called Q-value.

  Q: $(s_m, a_n) \rightarrow V_{mn}$……………………….. **(13)**

  For each couple (s_j, a_j) the associated value is initialized at random, the objective of the algorithm is that for each couple (s_j, a_j) improve the associated value V so that it converges towards the true value of the sum expected future rewards when the agent chooses the action a_j in the state s_j.
- The agent reinforces his knowledge by updating Q, the operation is repeated many times until the knowledge of the agent is sufficient to arrive at the solution.

Formally the algorithm is done in three steps:

1. In the state s_t the agent chooses an action $a_t$and observes the response of the environment which gives the reward r_t and the new state of the environment s_{t+1}
2. We compute a target composed of a reward r_t and a set of expected future rewards from t+1, approximated using current knowledge (Q) and a discount factor (λ).

Target Q value= r_{t+1}+ γmax_aQ(s_{t+1},a) - Q(s_t,a_t)...........................(14)

3. We update the function for the couple by mixing the knowledge resulting from the immediate experience (Target) and past knowledge , using Bellman's equation below :

$$\boxed{\text{New } Q(s,a)} = \boxed{Q(s,a)} + \boxed{\alpha}\boxed{[R(s,a)} + \boxed{\gamma}\boxed{\max Q'(s',a')} - \boxed{Q(s,a)}]$$

$$\mathbf{Q}(s_t, a_t) = \mathbf{Q}(s_t\ , a_t) + \alpha\ [r_{t+1} +\ \gamma max_a \mathbf{Q}(s_{t+1},\mathbf{a}) - \mathbf{Q}(s_t,a_t)]\text{.......}\mathbf{(15)}$$

Where:

- $Q(s_t,a_t)$: Current Q value.
- α : The learning rate determines how fast or slow the model will be learning.
- γ: Discount factor which defines the importance of the rewards we get now versus later in the episode.
- r_{t+1}: Reward.
- $MaxQ(s_{t+1},a)$: Maximum Expected Future Reward.

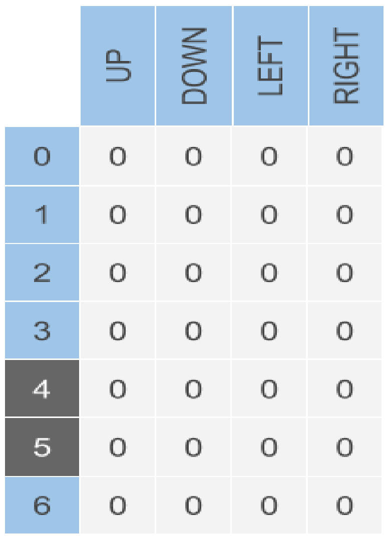
Q-Table:

We will encounter many solutions when we run our algorithm, and the agent will follow a number of different knowledge. How can we identify who is the best of them? This is accomplished by compiling our results into a table known as a Q-Table.

We can determine the appropriate knowledge of action for each environmental situation using a Q-Table. At each state, we apply the Bellman Equation to determine the future expected state and reward and save the results in a table for comparison with other states.

For an agent that needs to learn how to run, collect, and sit on command, let's develop a q-table. The steps involved in creating a q-table are:

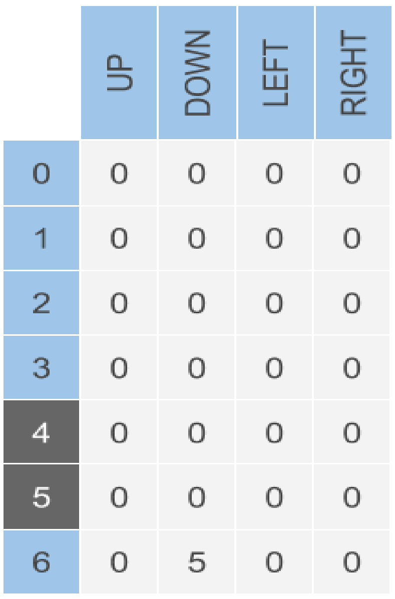
Step 1: Make a new Q-Table with all of its values set to 0. All states and rewards will start out with values of zero. Take a look at the Q-Table below, which demonstrates how it learns to do actions:



	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0

Figure 06 Initial Q-Table. [24]

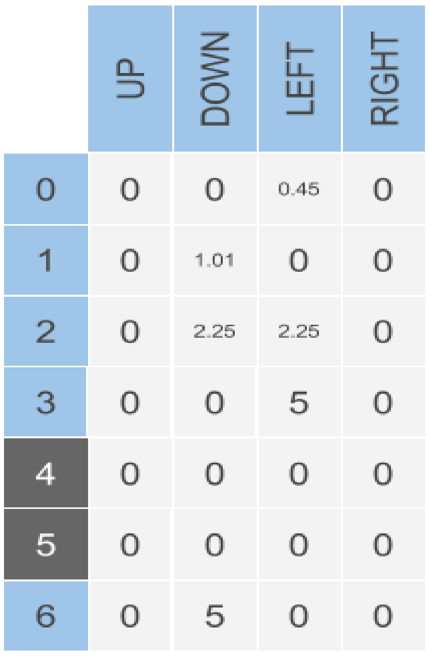
Step 2: Select a course of action and carry it out. Update the table's values.
We have not yet taken any further action. Let's imagine that we initially wanted the agent to sit, which it does. The new table will read:



	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0

Figure 07 Q-Table after making an action .[24]

Step 3: Determine the reward's value and use the Bellman Equation to determine the reward's Q-Value.



	UP	DOWN	LEFT	RIGHT
0	0	0	0.45	0
1	0	1.01	0	0
2	0	2.25	2.25	0
3	0	0	5	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0

Figure 08 Updating Q-Table.[24]

Step 4: Repeat this procedure up until the table is full or the end of an episode .The agent keeps acting, and for each action, it calculates the reward and Q-value and updates the table.

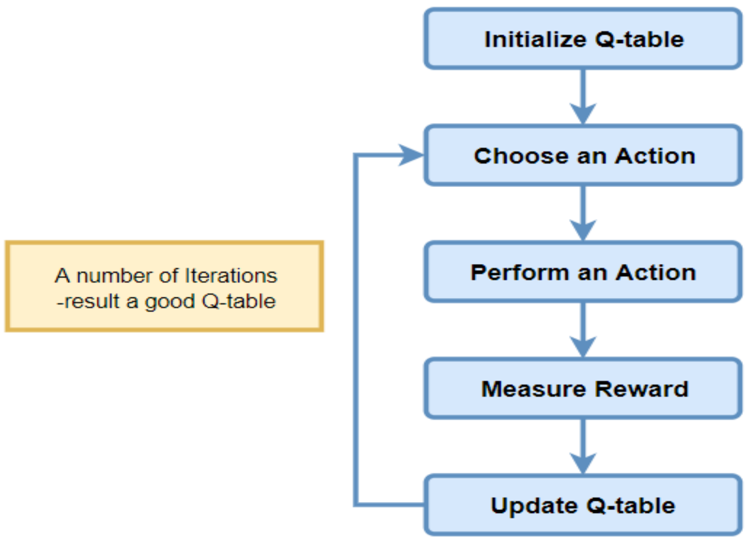


Figure 09 Final Q-Table. [24]

- **SARSA**

SARSA is an on-policy learning algorithm, which is deployed in state, Action, Reward, State, Action, and is a method based on policy values. We need a rule for updating values. It is shown in equation 4:

$$\mathbf{Q(s_t, a_t) = Q(s_t, a_t) + \alpha\, [r_{t+1} + \lambda Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]} \ldots\ldots\ldots\ldots\ldots\ldots \mathbf{(16)}$$

This is an improved Q learning algorithm where the target policy is the same as the behavior policy, two consecutive state-action pairs, and the immediate reward the agent receives when transitioning from the first state to the next state determines the updated Q value, So this method is called SARSA [26].

The difference between SARSA and Q-learning is the Q-value update rule that distinguishes SARSA from Q-learning. Thus, the time difference is calculated using the current action state and the next action state. This means that we need to know our policy's next action in order to perform the update step. This makes SARSA a policy-compliant algorithm because it is updated based on our current policy choices [26].

6. Deep Learning

6.1 Concept

Deep learning, a branch of machine learning that deals with neural networks. In deep learning, researchers attempted to duplicate the human neural network with an artificial neural network; in the deep learning model, the human neuron is referred to as a perceptron.

- **What is a Perceptron:**

The perceptron is an artificial neuron, a unit that belongs to an artificial neural network.

An artificial neuron is a simple Projection of a biological neuron. **Figure 10** where the artificial neuron mimics some function of the biological neuron such as learning and working in parallel [40].

The neuron is defined mathematically as a non linear and bounded algebraic function.

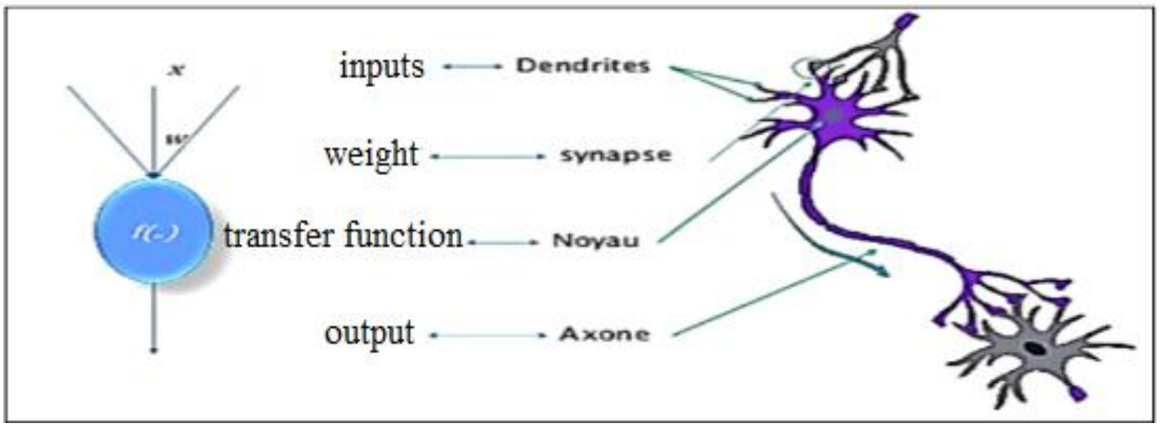


Figure 10: The artificial and biological neuron. [33]

- The artificial neuron functions in two phases[28]:
  - the first step is the preprocessing of the received data by calculating the potential Vj of neurons j by this function:

$$V_j = b_j + \sum_{i=1}^{n} w_{ij} x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(17)}$$

Where:

- w_{ij} represents the weight of the connection linking neuron j to input i.

-b_ja constant called bias considered as the weight of an input x_0.

- In the second step, a transfer function g called the activation function makes it possible to calculate the value of the internal state S_j of the neuron j, from $V_j$this value represents the neuron output.

S_j=g (V_j) =g ($\sum_{i=1}^{n} w_{ij}x_i$)…………………….. **(18)**

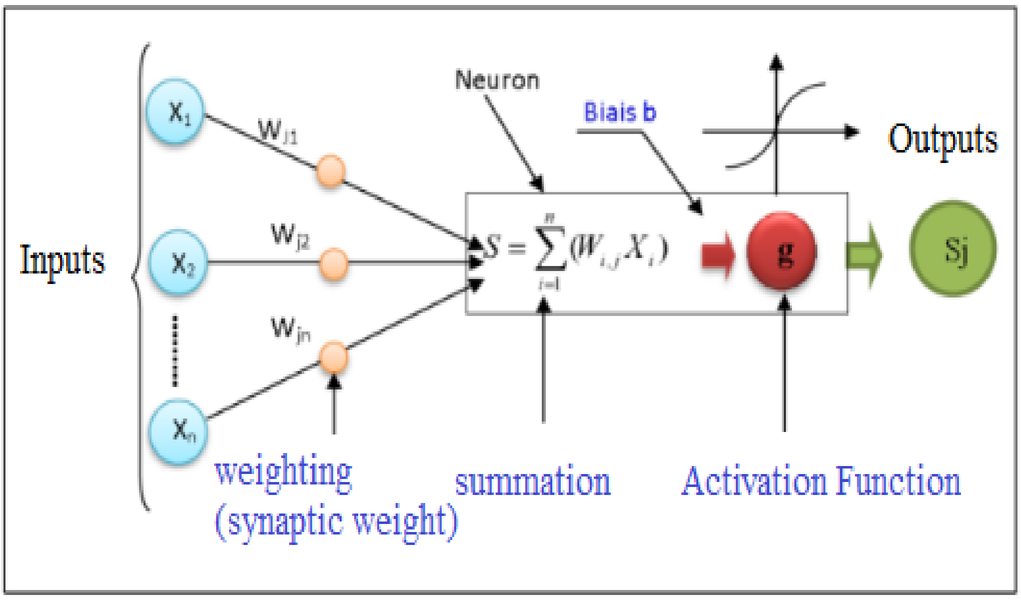


Figure 11 How a perceptron works [33].

- **Activation Function:**

The activation function or the transfer function allows to define the output of the neural network, there are several most used activation functions [33] which are showed in the **Figure 12**:

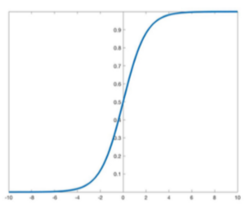
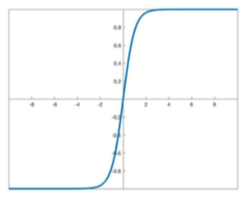
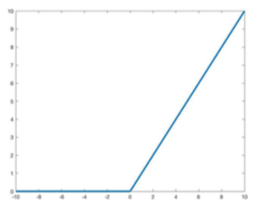
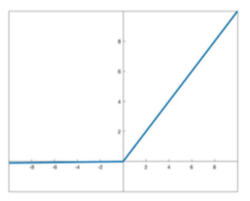
Activation function	Equation	Graph
Sigmoid	$S(x) = \frac{1}{1+e^{-x}}$	
Tanh	$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	
ReLU	$RELU(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$	

Figure 12 common activation functions. [33]

6.2 Connexion Between Neurons:

The connections between the neurons that make up the network describe the topology of the model, so we defined the most common type of neural nets:

6.2.1. Feed Forward Network

Feed-Forward refers to the procedure of data processing by the neural network, feed-forward means that the data passes through the network from input to output without backtracking information. That is to say if we move in the network from any neuron following the connections we cannot return to the starting neuron.

In the family of forward propagation there are two types of networks seen in the following figures:

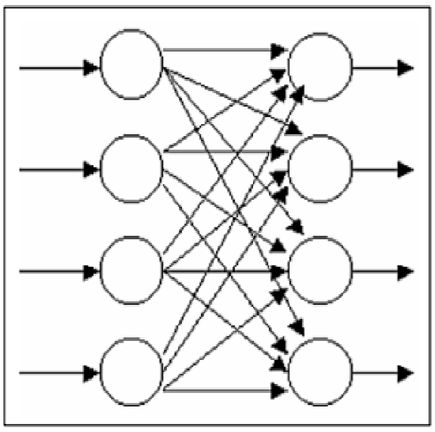


Figure 13 Total connection network. [27]

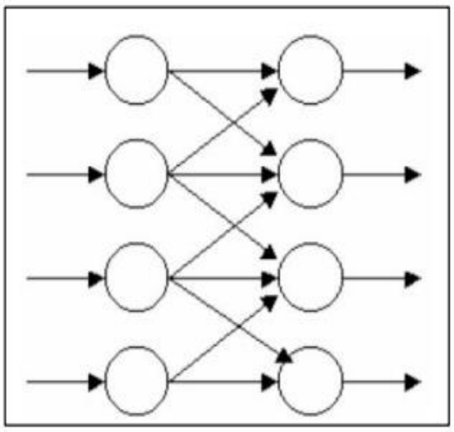


Figure 14 Partially connected networks. [27]

6.2.2. Recurrent Neural Network

Recurrent neural networks (feedback network) process information in cycles These cycles allow the network to process information several times by sending it back to the network each time, these are networks with several internal loops their output at a given time depends on inputs at the same times, such that the strength of recurrent neural networks appear in their ability to take information into account follows the recurrence of the processing of the same information where there are recurrent neural networks composed of one or more layers and other multi-layer network.

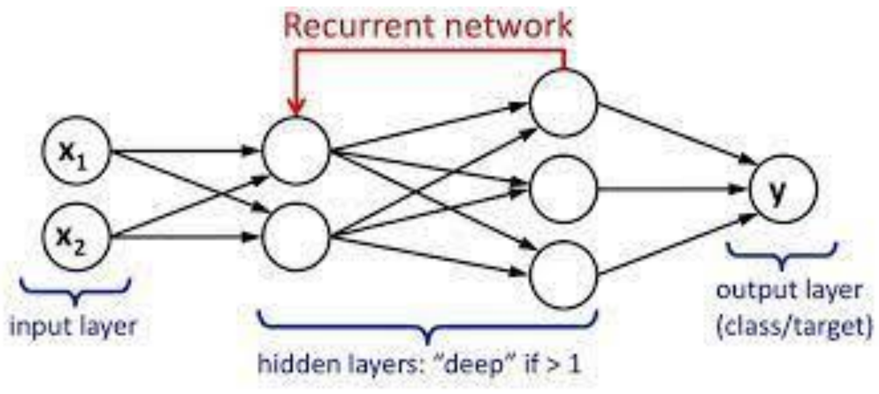


Figure 15 Recurrent Neural Network.[27]

6.2.3. Deep Neural Network:

An artificial neural network (ANN) with several layers between the input and output layers is known as a deep neural network (DNN). Where the high number of layers should be a source of problem where from a number of layers the network artificial neural network was not able to learn properly so the solution to this problem is the deep neural network which has more and more endowed with multiple layers and able to learn

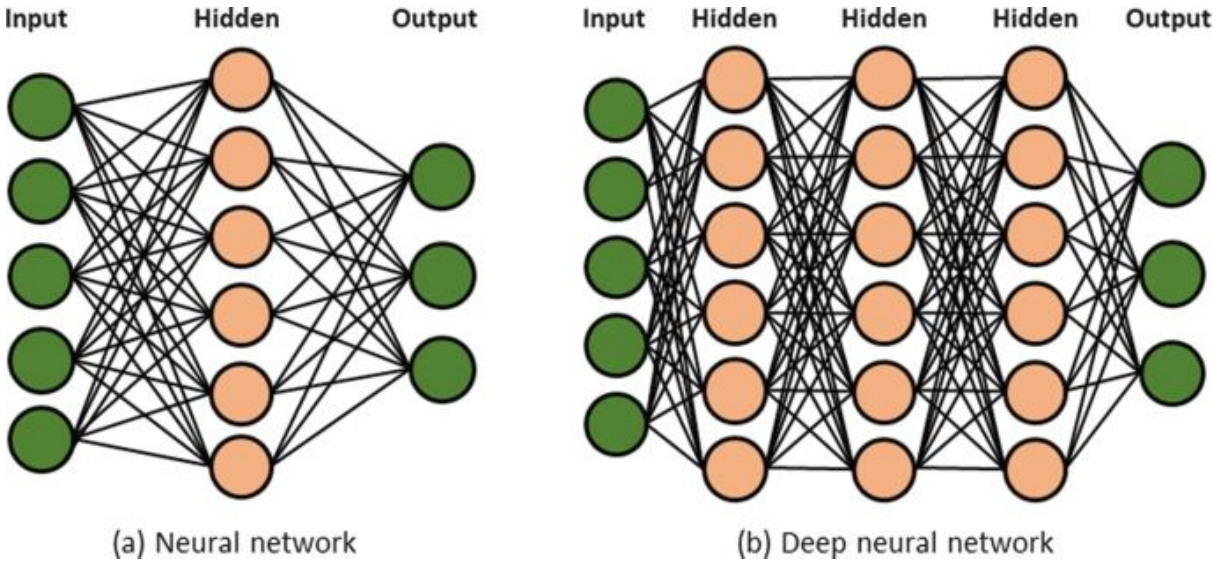


Figure16 The difference between Artificial Neural network and Deep neural network.[28]

In deep neural network you can use all the formulas and technique to shape each layer for example a deep neural network can have three fully connected blocks spotted at the input and then take a function from the output of the first block and another function of the second block but in artificial neural networks you are limited to using fully connected layers. [35]
In other words, the deep neural network where we construct a neural network by connecting these perceptron units together; it comprises three components:

Input layer ->Hidden layers ->Output layer

The input layer: is the layer responsible for receiving input such that the layer taking the input is performing calculations through the neurons and then the output is transmitted to the following layers**.**

The hidden layer: The input and output layers are separated by the hidden layers. It is the heart of the perceptron, these layers are invisible it is private for neural networks where each layer contains the same total number of neurons

The output layer: is primarily responsible for producing the final output results.

- In the hidden layers we can have:

  • **The CONVOLUTIONAL layer** It is about detecting features and applying filters, and it will generate the required output after testing on a database by computing a dot product between the weight and region that are associated to the input volumes.

  • **The RELU layer** is utilized to perform an element-by-element activation function, in which we replace all negative values in the filtered picture with zero. When the node input surpasses a particular threshold, this function is activated. As a result, when the input is less than zero, the output is also zero. When the input surpasses a particular threshold, however, the dependent variable and the input have a linear relationship. This implies it can increase the speed of a training data set in a deep neural network quicker than other activation functions, avoiding summing with zero.

  • **POOL layer** is used to lower the size and volume of the width and height, and keeping the important features.

  • **The FULLY CONNECTED layer** at the end of the convolutional neural network, the class scores will be calculated and build all needed connections.

  • **FLATTENING layer** is used to convert to 1 dimension array.

7. Deep Reinforcement learning:

A recent area of study in the field of machine learning is deep reinforcement learning. In addition to being the driving force behind recent advances in issues like computer vision, machine translation, and time series prediction, neural networks can also be used with reinforcement learning methods to produce amazing results [39].

Deep learning-enhanced reinforcement algorithms have the ability to defeat human professionals at a variety of video games as well as world champions. Even though it might seem insignificant, it represents a significant advancement over their prior successes, and technology is developing quickly [39].

Like humans, our agents develop the best long-term strategies for themselves in order to attain the greatest rewards, this concept of learning by trial-and-error only through rewards is known as Reinforcement Learning (RL).

Our agents generate and learn their own knowledge directly from raw inputs, just like humans do. This is made possible via deep learning of neural networks. The foundation of Deep

Reinforcement Learning is the training of deep neural networks to approximation the optimum policy and the ideal value functions V, Q.

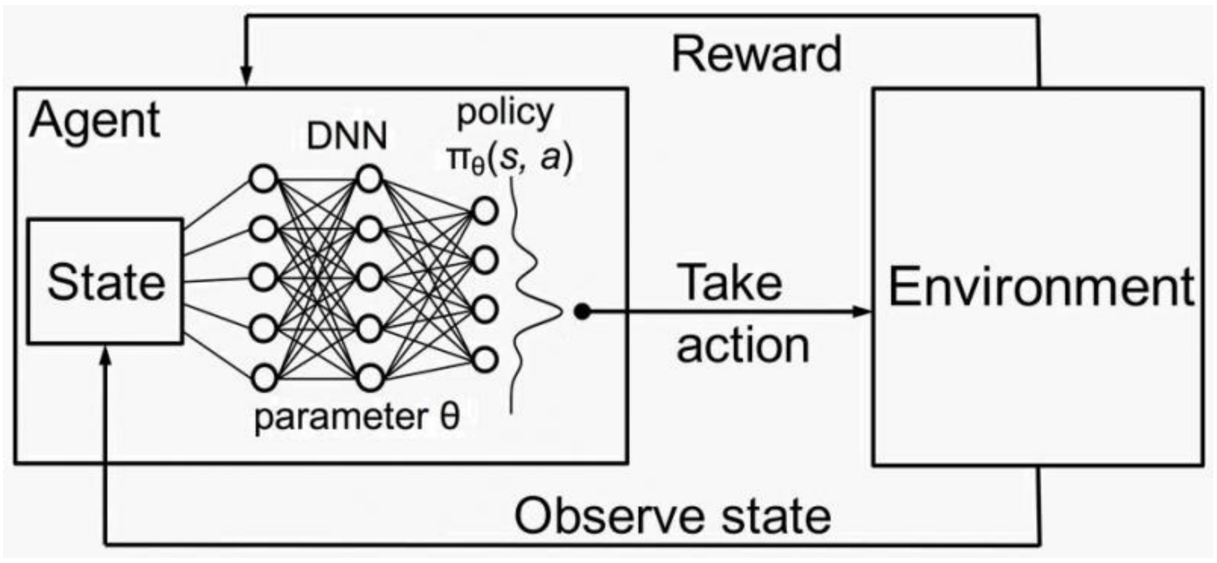


Figure 17 Deep Reinforcement Learning. [39]

We are combining two things where RL sets the objective, and DL provides the mechanism, the way to express problems, and the way to solve them. Our goal is to create a single agent that can handle any human-level work. RL and DL produce general intelligence, or the ability to solve many complex problems.

So how can RL and DL be combined?

- Deep neural networks can be used to represent
    - Function of values
    - Policy
    - Model
- Improve the loss function.

The general working of a Neural Network is to:

- Set up your primary and target neural networks.
- select an action.
- update the network weights.

8. Conclusion

This chapter presented learning and discussed the distinctions between deep learning and classical neural networks. To be able to explain the choice of the structure used in the following chapter, we have presented the various deep neural network structures and their principles of use.

Chapter 3: Proposed Approach

1. Introduction

Instead of using traditional algorithms like UApriori, a new approach was proposed in this chapter to extract frequent itemsets and high utility itemsets from uncertain data. We provide a strategy to extracting uncertain patterns based on reinforcement learning called "U-PMRL" that employs a deep neural network called "Deep Q Network".

2. Motivation Attributed to Using a Deep Model

Due to the extensive search space, extracting the itemset is difficult. The researchers have created a variety of data structures and algorithms including the UApriori algorithm and the UP-Growth algorithm to trim the search space because the naïve [40] approach is impractical to analyzing each of these item set. These itemsets extraction approaches are inflexible, nevertheless, the search space grows exponentially as the number of elements increases because the execution time of these algorithms deteriorates as the dataset enlarges.

The main concept is derived from how a human searches a dataset for a particular type of itemsets. Humans most likely start by quickly skimming the dataset to gain a general understanding of which elements appear to be relevant (or unimportant) for the target type, which items are related to one another, and so forth. Then, he or she creates a "set of objects" made up of a few crucial elements and determines whether or not it matches the goal type. After that, a person updates the collection of items by adding or removing items to construct a new itemset based on knowledge gained from previous search experiences.

Given its recent success, reinforcement learning which is based on behavioural psychology, we suggest an itemsets extraction approach that is based on RL.RL is used in Itemsets Mining to train an artificial agent to extract a collection of patterns from an uncertain dataset, what we called "U-PMRL" for Uncertain Pattern Mining based on Reinforcement Learning

3. Extraction of Uncertain Pattern Based on Reinforcement Learning (U-PMRL)

When we start the work on this project there was none research that use RL for mining patterns from data, but later, KAZUMA and KIMIAKI [35] have published a paper which deals with the same problem in very similar manner. As it is the first work in the literature in our problematic context, we have been inspired by the approach proposed in this paper. However, in our case, we address the problem of mining itemsets from uncertain data.

In fact, **U-PMRL** (for Uncertain Pattern Mining based on Reinforcement Learning) is an approach that uses reinforcement learning to train a machine learning model extracting itemsets. A computer-generated agent interacts with an uncertain environment to adaptively learn the optimal policy of action. The agent then is trained to extract a set of items from an uncertain dataset in the following manner:

- First, the agent performs an action to update a set of items by adding or removing an item.
- Then, the environment represented by the probabilistic dataset generates a rewardr_k **Figure18** which expresses the relevance of the set of elements updated by the agent, such that the reward value is defined as follows:

- **Case 1:** The agent obtains a reward of -1,r_k=-1 if the set of items resulting from the action r_k in state $s_k$does not exist in the probabilistic database.
- **Case 2:** The agent obtains a reward of 0,r_k =0 if the calculated Expected support value (ExpSupp) of an itemset X as a result of the action $a_k$in state sk is less than a quarter of a predetermined threshold, i.e.

  (ExpSupp(X) ≤ ε /4) the same for the Probabilistic support

  Where j is the number of transactions :

  ExpSupp(X)=$\sum_{j=1}^{n} P\left(X\,,t_j\right) = \sum_{j=1}^{n}(\prod_{x\in X} P(x,\ t_j)$

  ProbSupp(X)=$P_i(X)$=$\sum_{\substack{S\sqsubseteq T\\|S|=i}}(\prod_{t\in S} P(X \sqsubseteq t)\prod_{t\epsilon T-S} 1 - P(X \sqsubseteq t)$
- **Case 3:** ExpSupp(X) ≥ ε/4 (ProbSupp(X)) is the condition of this case which is then divided into four sub-cases, if $\varepsilon/4$ ≤ExpSupp(X) ≤ $\varepsilon/2$ (ProbSupp(X)) then the agent receives a reward equal to 1 else, if $\varepsilon/4$ ≤ExpSupp(X) ≤ $3\varepsilon/4$ (ProbSupp(X)) then the agent receives a reward equal to 2 else, if $3\varepsilon/4$ ≤ ExpSupp(X)≤ ε ((ProbSupp(X)) then the agent receives a reward equal to 3 else, if ExpSupp(X)≥ ε (ProbSupp(X)) then the agent receives a reward equal to 4.
- **Case 4:** Two situations characterize this case. If the set of itemset X matches these two requirements, then offers a very high reward of 100. The first tests if the ExpSupp(X),(ProbSupp(X)) is greater than the threshold, i.e. (ExpSupp(X) ≥ ε), and the second checks if the set of itemset X has not yet been extracted in the episode.

Therefore, **U-PMRL** gathers a substantial amount of trial-and-error steps in which the agent occasionally succeeded or failed to create the set of target elements. The agent is trained to have the best adding or deleting policies by analyzing these trial and error steps. For this we employ the DeepQ-learning algorithm.

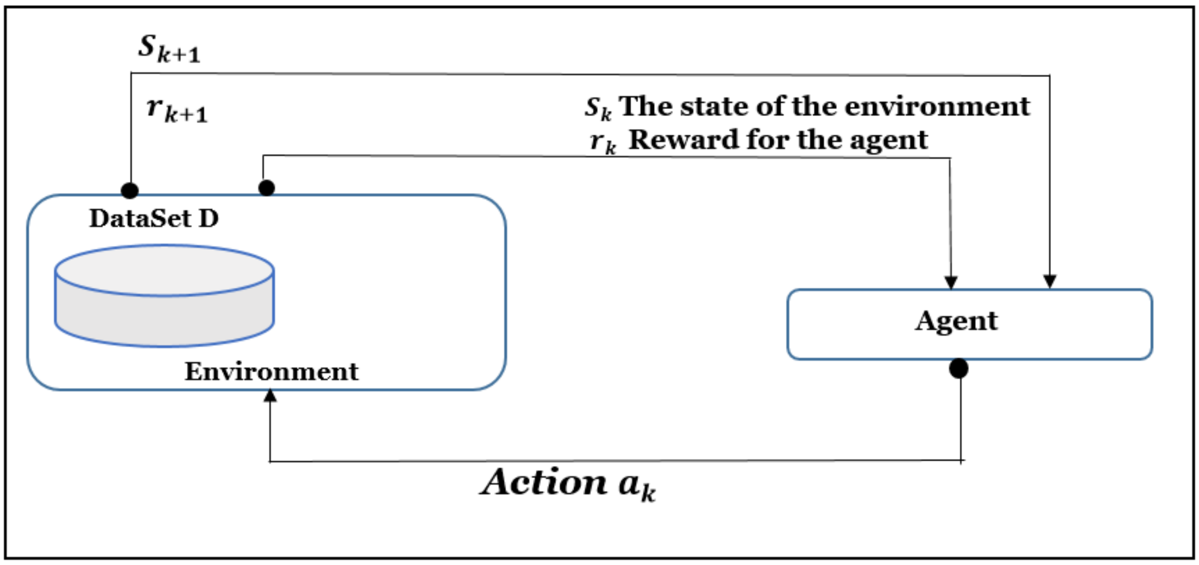

s_{k+1}
r_{k+1}
s_k The state of the environment
r_k Reward for the agent
DataSet D
Environment
Agent
Action a_k

Figure 18 The interaction cycle between the agent and the environment.

4. The architecture of the Deep Q-learning algorithm

Although Q-Learning is beneficial and too basic for our problems, our project is too sophisticated for Q-Learning to handle.

Following those investigations and research, we conclude that deep learning works best for complicated tasks but requires a lot of data, whereas Q learning works best for basic projects. As a result, we combined the two to create Deep Q learning (DQN).

We employ the Q-learning algorithm, which develops an agent with a function Q that takes an action a and a state s of the environment as inputs and outputs the quality of the action $a$at s.

We utilize a Neural Network that determines the Q-values for each action depending on a state. We use a Deep Neural Network that gets the state as input, and produces different Q values for each action.

To be in the Deep Q Learning, We utilize a Neural Network to approximately determine the Q-values for each action depending on a state. We use a Deep Neural Network that gets the state as input, and produces different Q values for each action.

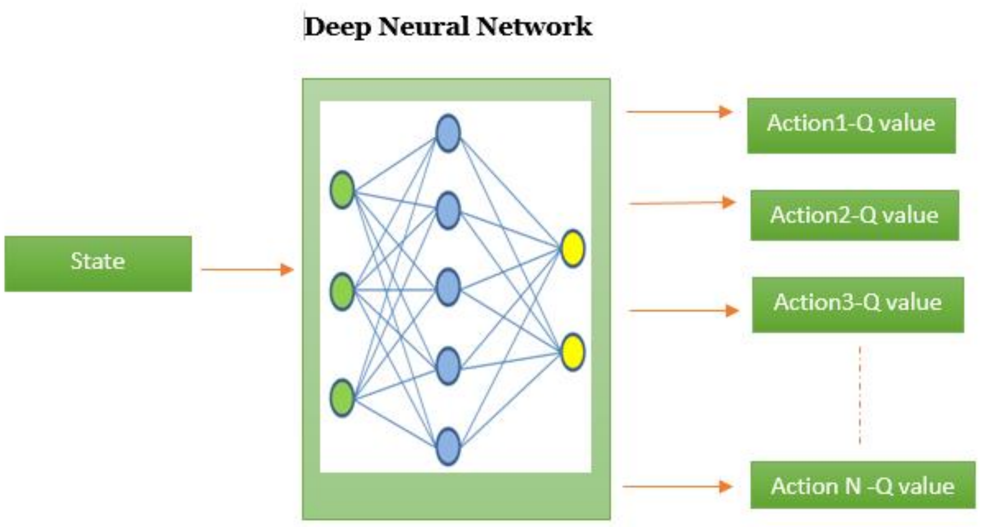


Figure19 Structure of Deep Q Learning

Due to the large number of state-action combinations, it is not practical to directly construct$Q(s,a)$. For this reason, we approach $Q(s,a)$ by a neural network "Deep Q Network" (DQN), which is defined by a set of parameters. $Q(s,a)$ indicates the maximum number of target itemsets after taking an action(a) in the state(s). $Q(s,a)$is parameterized by Q(s, a,θ) and our objective is to optimize DQN so that the chosen actions produce numerous itemsets.

The DQN neural network architecture for extracting frequent itemsets and high utility itemsets respectively are common with respect to the input and output layers such that the input layer accepts a state vector an M dimension $s_k = s_{k,1}, \dots \dots, s_{k,m}$(k step) where $1 \leq m \leq M$ represents a utility of changing of the inclusion of the element in the set of elements, and the output layer generates a vector$q_k = q_{k,1}, \dots \dots, q_{k,m+1}$ which denotes the quality of the action to change the inclusion of an element and the one for the random bit-vector initialisation.

So, the DQN consists of three layers:

- Input Layer.
- Hidden Layer which contain 3 blocks and every block contain a fully connected layer and batch normalisation layer, leaky relu layer.
- Output Layer.

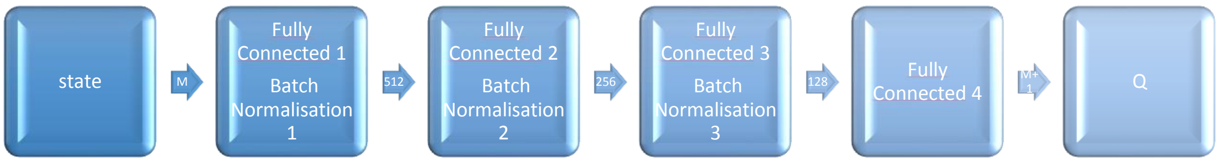
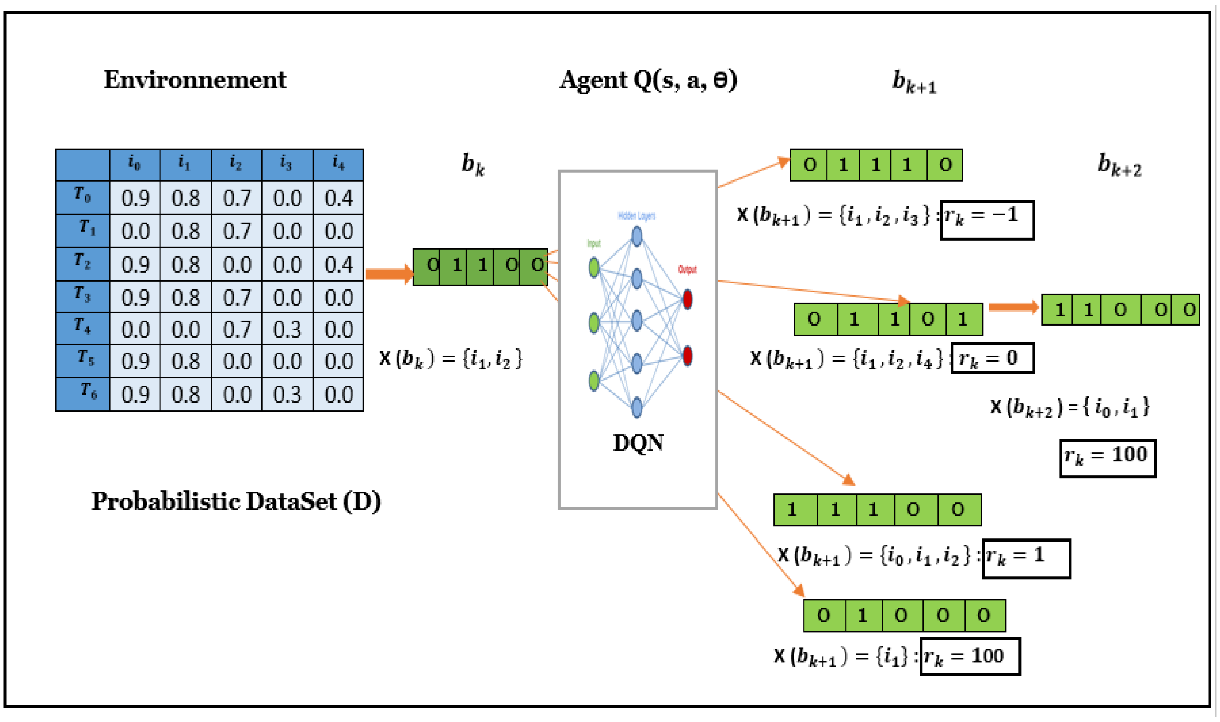


Figure 20 DQN architectures

5. Presentation of the Proposed Approach

5.1 Case of mining uncertain frequent itemsets

We provide the following graphic to help you understand U-PMRL and how the agent Q(s, a,θ) trains to extract frequent itemsets from the probabilistic database.


Environnement
Agent Q(s, a, ϴ)
b_{k+1}
b_k
b_{k+2}
0 1 1 0 0
X (b_k) = {i_1, i_2}
Hidden Layers
Input
Output
DQN
Probabilistic DataSet (D)
0 1 1 1 0
X (b_{k+1}) = {i_1, i_2, i_3} : $r_k = -1$
0 1 1 0 1
X (b_{k+1}) = {i_1, i_2, i_4} : $r_k = 0$
1 1 0 0 0
X (b_{k+2}) = { i_0, i_1 }
$r_k = 100$
1 1 1 0 0
X (b_{k+1}) = {i_0, i_1, i_2} : $r_k = 1$
0 1 0 0 0
X (b_{k+1}) = {i_1} : $r_k = 100$

	i_0	i_1	i_2	i_3	i_4
T_0	0.9	0.8	0.7	0.0	0.4
T_1	0.0	0.8	0.7	0.0	0.0
T_2	0.9	0.8	0.0	0.0	0.4
T_3	0.9	0.8	0.7	0.0	0.0
T_4	0.0	0.0	0.7	0.3	0.0
T_5	0.9	0.8	0.0	0.0	0.0
T_6	0.9	0.8	0.0	0.3	0.0

Figure 21 How agent train to extract frequent itemsets.

The figure represents the steps k, k+1, k+2 to create an agent on uncertain dataset D, where b_k represents the M-dimensional binary vector where$b_{k,m} \epsilon \{0,1\}$ where $1 \leq m \leq$M which represents the inclusion of the item i_m in D, and we describe the itemsets defined by b_k by X(b_k).

- The vectorb_k =[0,1,1,0,0] shown in Figure 3 is randomly generated at the k th step thus define the itemset $\{i_1, i_2\}$, at state k+1 the vectorb_{k+1} =[0,1,1,1,0] shown in

Figure 21 resulting by the action of changing the inclusion of i_3 in X(b_k) and according to the definition of the reward r_k=-1 because X(b_{k+1}) defined by b_{k+1} does not exist in D consequently θ is updated such that Q(s,a,θ) produces a very small value for the action of changing the inclusion of i_3 in the s_k calculated state from b_k more simply the agent becomes not to select this action for b_k.

- In the second case based on b_{k+1} =[0,1,1,0,1] obtained by the action of changing the inclusion of i_4 in X(b_k), since X(b_{k+1})={i_1, i_2, i_4} appearing once in D the agent receives the reward r_k= 0 because ExpSupp(X(b_{k+1}))≤ $\varepsilon/4$ knowing that ε= 0.9
- For the third case changes the inclusion of i_0 in X(b_k),thus produces X(b_{k+1})={i_0 , i_1 , i_2} which appear twice in D since this case is classified as the first sub-case of the **Case 3** the agent receives the reward r_k=1 consequently Q(s,a,θ) produces a high value for the action to modify the inclusion of i_0 in the state s_k which allows the agent to optionally select this action .
- The 4th case triggered by the action of changing the inclusion of i_2 in X(b_k) and the result X(b_{k+1}) =i_1 appears 6 times in D is extracted as FI, the reward r_k =100 guides Q(s,a,θ) to produce a high value for this action which makes the agent likely to select this action.

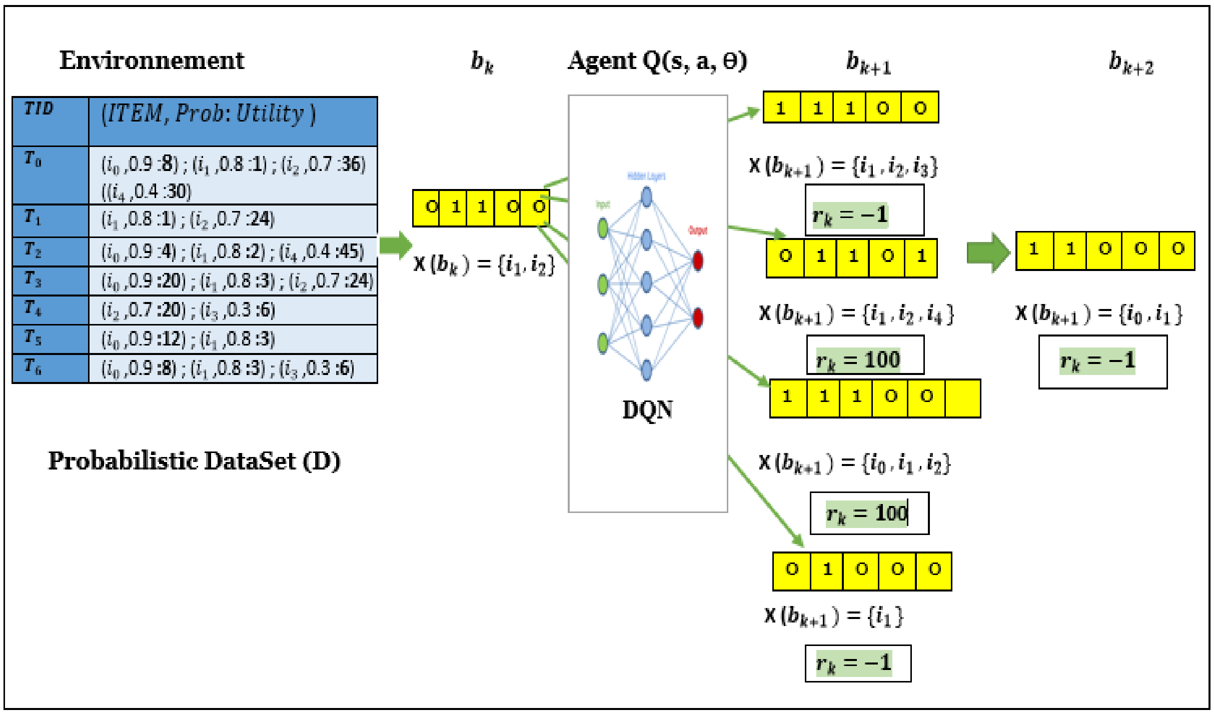
  The training of Q(s, a,θ) considers not only the reward r_k at the kth step but also at future steps as illustrated in the last case caused by the action of modifying the inclusion of $i_2$in X(b_{k+1}) and the result X(b_{k+2})={i_0, i_1} appears 5 times in D is extracted as an IF, the r_k=100 obviously guides Q(s,a,θ) to generate a high value for the action modification of the inclusion of i_3 to s_{k+1}calculated fromb_{k+1}.

In this way, in the U-PMRL method an agent is trained to select an action that leads to the extraction of an FI

5.2 Extracting Uncertain High Utility Itemsets

After demonstrating the extraction of frequent itemsets, we used our method to extract high utility itemsets from a uncertain database, even though the extraction of high utility itemsets from uncertain data is frequently seen in real-world applications where the utility of the item in a transaction is represented by the multiplication of the profit by the quantity of the item itself.

The graphic below serves as an illustration of how the agent trained to extract HUI from uncertain database.



TID	$(ITEM, Prob: Utility)$
T_0	$(i_0, 0.9:8)$; $(i_1, 0.8:1)$; $(i_2, 0.7:36)$ $((i_4, 0.4:30)$
T_1	$(i_1, 0.8:1)$; $(i_2, 0.7:24)$
T_2	$(i_0, 0.9:4)$; $(i_1, 0.8:2)$; $(i_4, 0.4:45)$
T_3	$(i_0, 0.9:20)$; $(i_1, 0.8:3)$; $(i_2, 0.7:24)$
T_4	$(i_2, 0.7:20)$; $(i_3, 0.3:6)$
T_5	$(i_0, 0.9:12)$; $(i_1, 0.8:3)$
T_6	$(i_0, 0.9:8)$; $(i_1, 0.8:3)$; $(i_3, 0.3:6)$

Figure 22 How agent train to extract high utility itemsets.

The steps shown in the picture is k, k+1, k+2 which is used to create an agent on the probabilistic dataset D. Here, b_k stands for the binary vector that defines whether an item i_m is included in D where $b_{k,m} \epsilon \{0, 1\}$, and we describe the item sets given by b_k by X(b_k).

- In step k , the vector b_k=[0,1,1,0,0] is randomly generated at the k-th step thus defining the itemset X(b_k)={i_1,i_2} , at the state k+1 the vector b_{k+1}=[0,1,1,1,0] shown in the **Figure 22** resulting by the action of changing the inclusion of i_3 in X(b_k) and according to the definition of the reward r_k=-1 because X(b_{k+1}) defined by b_{k+1} does not exist in D consequently θ is updated such that Q(s,a,θ) produces a very small value for the action of changing the inclusion of $i_3$in the s_k calculated state from b_k more simply the agent becomes not to select this action for b_k.
- The second case on b_{k+1}=[0,1,1,0,1] obtained by the action of changing the inclusion of $i_4$in X(b_k) the agent receive the reward r_k=100 because according to the definition of the utility of an itemset X(b_{k+1})= {i_1,i_2,i_4} equal to **67**, X(b_{k+1})is considered as high utility itemset because U(X(b_{k+1}),T_q)≥ ε*TU consequently Q(s,a,θ) produces a high value for the action to modify the inclusion of i_2 in the state s_k which allows the agent to optionally select the action.
- For the third case triggered by the action of changing the inclusion of i_0 in X(b_k)this produces the X(b_{k+1})={i_0 ,i_1 ,i_2 } witch the utility value of

X(b_{k+1}) equal to **92** the agent receive the reward r_k =100 because U(X(b_{k+1}),T_q)≥ $\varepsilon * Tu$, the Q(s,a,θ) produces a high value for the action to change the inclusion of i_0 in the state s_k which allows the agent to select this action .

- The 4th case triggered by the action of changing the inclusion of i_2 in X(b_k) and the result X(b_{k+1})={ i_1 } appears 6 times in D is not extracted as HUI because U(X(b_{k+1}))=U({i_1 })=13≤ $\varepsilon * Tu$, the reward r_k =-1 which makes the agent not select this action,consequently θ is updated such that Q(s,a,θ) produces a very small value for the action of changing the inclusion of i_3 in the state s_k more simply the agent becomes not to select this action for b_k .
- In the last case caused by the action of modifying the inclusion of i_2 in X(b_{k+1})and the result X(b_{k+2})={i_0 , i_1 } is not extracted as HUI, sach that the agent receive reward r_k = -1 obviously It is clear that this causes Q(s,a,θ) to produce a small value for the action modification of adding i_2 to s_{k+1} determined from b_{k+1} .

In this way, the agent is trained to select an action that leads to the extraction of an HUI.

- **Pseudo algorithm**

The proposed algorithm named **U-PMRL** is presented as follows:

Algorithm: Uncertain Pattern Mining Based on Reinforcement learning (**U-PMRL**).

Input DataSetD, Expected Support ExpSupp(X) , threshold ε;

Output a set ITEMS containing itemsets meeting ExpSupp(X)≥ ε , $ProbSupp(X) \geq \varepsilon$, $U(X,T) \geq TU \times \varepsilon$.

```
Initialize  Q(s,a,$\theta$) with parameter installation
Initialize a target network as Q(s,a,$\theta$)=Q(s,a,$\theta^-$)
Initialize a replay memory as  $P$ ← {} ;
Reduction the search space by discarding items which are less than threshold(have no chance to be in itemsets)
X ← {} ;
For   e=1…….E   Do
        Randomly initialize bit vector $b_1$
        Generate the first state S based on b1 and D
        For   K=1……K   Do
                calculate $q_k$by introducing $s_k$ in Q(s,a,$\theta$)
                Decide an action $a_k$ from the $q_k$
                Update from $b_k$ to $b_{k+1}$ by $a_k$
                **If** ExpSupp( X($b_{k+1}$))≥ $\varepsilon$ **then**
        /*Expected support, Probabilistic support or Utility $u(X,Tq) \geq TU \times \varepsilon$*/
                ITEMS ← $ITEMS \cup$ X($b_{k+1}$)
                **Endif;**
                calculate the reward $r_k$ based on $b_{k+1}$ and D
                calculates $s_{k+1}$ as a function of $b_{k+1}$ and D
                $P \leftarrow P \cup \{ (s_k, a_k, r_k, s_{k+1}) \}$ ;
                Update $\theta$ of Q(s, a,$\theta$) using randomly drawn experiments from P
        **EndFor;**
        Update $\theta^-$ of Q(s,a,$\theta^-$)**(every episodes)**
**EndFor;**
**return** ITEMSETS
```

Formally, the agent was trained in e episodes, each episode contain k steps, and at each step the agent received state s and count quality q to decide which action a to choose for update b_k (the bit vector) into b_{k+1} . Then, if the item set satisfy the condition ExpSupp(X(b_{k+1}))≥ ε , the environment computes its reward and goes to the next state S_{k+1}. Then, it will store the tuple (s_k, a_k , r_k, s_{k+1}) in the replay memory (p) as an experience of agent. Consequently, the parameter θ will be updated.

Finally, the parameter θ^- of the network will be updated each episodes.

6. Conclusion

In this chapter we presented our approach for extracting itemsets based on a deep reinforcement learning model, we also presented the steps to take to train a neural network as well as our model.

The following chapter is devoted to demonstrating the effectiveness of our method by detailed experimental study on various datasets.

Chapter4: Testing and Experimentation

1. Introduction

We'll talk in this chapter about the experimental section of our strategy after we've detailed our answer. The working environment and tools that we utilized will be shown in the first section. Then we'll present our test set, and then we'll show the results that we generated. Consequently the performance of our proposed approach.

2. Working Environment

2.1 Hardware Environment:

- An HP computer with the following characteristics:
  - 8,00 Go RAM.
  - Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz 2.50 GHz.
  - Operating system (OS): Windows 10 - 64bit.
- A DELL computer with the following characteristics:
  - 4,00 Go RAM.
  - Intel(R) Celeron(R) CPU N3060 @ 1.60GHz 1.60 GHz .
  - Operating system (OS): Windows 10 - 64bit.

2.2 Software Environnent

- **Python 3.7:** Python was developed by Guido van in 1990 and is one of the most widely used programming languages. It is strong and easy to learn, has dynamic typing, and is great for scripting and quick construction of applications in a variety of fields. It also supports high-level data structures. Python syntax allows programmers to code in fewer steps than Java or C++, making object-oriented programming clear and efficient. Because of its many programming paradigms, the language is extensively used in big businesses. It contains a vast and comprehensive standard library, as well as automated memory management and dynamic operation.

- **Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows you to write and share computer code, visualizations, equations, and text. It can be used to analyze a collection of texts, clean them up, and alter them. It can also be used for statistical modeling and automatic learning.

  Jupyter Notebook is a two-in-one tool:

  - Jupyter Notebook Documents are documents created by the Jupyter notebook application that includes both enhanced text (paragraph, equation) written in

LATEX and computer code. The notebook documents are both human-readable and executable documents that may be used to do data analysis.

- Application Jupyter Notebook It is a client-server application that allows you to change and execute notebook documents via a web browser, in addition to displaying, running, and changing documents. The Jupyter notebook application can run without an internet connection or on a server with an internet connection.

The library used :

- **Pandas:** It is a high-level tool for python analysis, an open-source software library created for data manipulation and analysis in python that is powerful, flexible, and simple to use. Pandas is built on "DataFrames", which are two-dimensional data arrays.
- **Tensor Flow:** is a Python-compatible open source library designed by the Google Brain team for high-performance numerical computation and machine learning. Its flexible architecture allows computation to be deployed across a number of platforms (CPU, GPU) and from PCs to server clusters, simplifies the data acquisition, prediction generation, machine learning model training, and future results improvement processes
- **NumPy:** Numerical Python's abbreviation is this library is extremely helpful for doing mathematical and static operations in Python, and it works flawlessly with many matrices and multidimensional tables.
- **sklearn:** is the abbreviation of scikit-learn is undoubtedly python's most helpful machine learning library. classification, regression, clustering, and dimensionality reduction are just a few of the useful functions in the sklearn library for machine learning and statistical modeling.
- **gym:** itis an open-source Python library for creating and comparing reinforcement learning algorithms by offering a standard API to communicate among learning algorithms and the environments. This python library provides us with a large number of test environments in which we can perform on our RL agent's algorithms, as well as shared interfaces for writing and testing general algorithms.
- **torch:** It is a module that allows replacing NumPy to use the power of GPUs, to provide a platform for deep learning. The notion of tensors is to replace arrays of

NumPy and allows: tensors creation, operations on tensors, resizing, indexing, the gateway with NumPy, Multiplication of 2 tensors..etc.

And we use a torch.nn base class that can be used to wrap parameters, functions, and layers in the torch.nn modules. Any deep learning model is developed using the torch.nn module which is a base class for all neural network modules.

3. Data Preprocessing

3.1 The used data

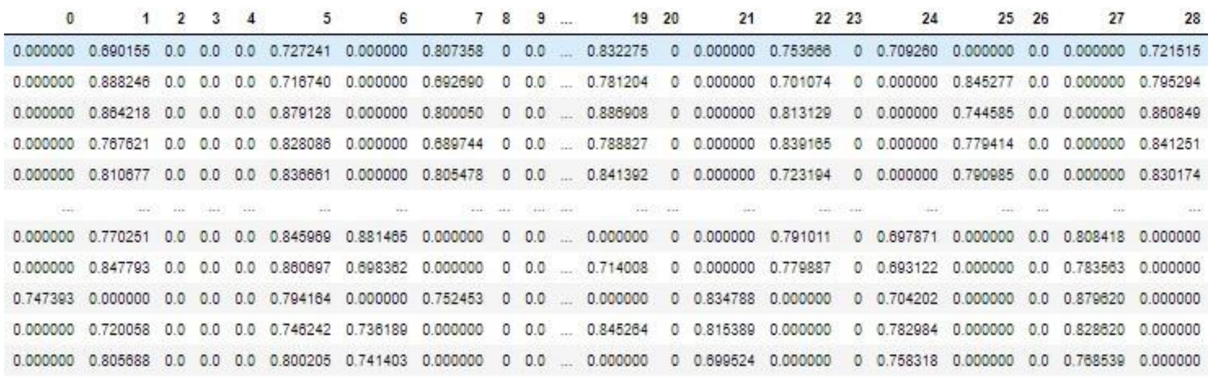
The Machine Learning Repository (UCI) is a collection of domain theory databases and data generators that the machine learning community uses to analyze machine learning methods.

Its datasets are appropriate for the extraction of frequent patterns itemsets and high utility itemsets, among the data sets are 'UBRecordLink', 'UBSkin'.

Data production is one of the first things to consider before training the model. For our model, we dealt with data items with uncertain values (probabilities).

Therefore, we convert those datasets from certain to uncertain values as shown in the Figure below, where we used the normal law to generate probability between 0 and 1 for each item of each transaction.

In addition The UBRecordlink data has 29 items and UBSkin data has 11 items, and for the number of transactions in our test we have 200-1000 transactions.



0	1	2	3	4	5	6	7	8	9	...	19	20	21	22	23	24	25	26	27	28
0.000000	0.690155	0.0	0.0	0.0	0.727241	0.000000	0.807358	0	0.0	...	0.832275	0	0.000000	0.753666	0	0.709260	0.000000	0.0	0.000000	0.721515
0.000000	0.888246	0.0	0.0	0.0	0.716740	0.000000	0.692690	0	0.0	...	0.781204	0	0.000000	0.701074	0	0.000000	0.845277	0.0	0.000000	0.795294
0.000000	0.864218	0.0	0.0	0.0	0.879128	0.000000	0.800050	0	0.0	...	0.886908	0	0.000000	0.813129	0	0.000000	0.744585	0.0	0.000000	0.860849
0.000000	0.767621	0.0	0.0	0.0	0.828086	0.000000	0.689744	0	0.0	...	0.788827	0	0.000000	0.839165	0	0.000000	0.779414	0.0	0.000000	0.841251
0.000000	0.810677	0.0	0.0	0.0	0.836661	0.000000	0.805478	0	0.0	...	0.841392	0	0.000000	0.723194	0	0.000000	0.790985	0.0	0.000000	0.830174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.000000	0.770251	0.0	0.0	0.0	0.845969	0.881465	0.000000	0	0.0	...	0.000000	0	0.000000	0.791011	0	0.697871	0.000000	0.0	0.808418	0.000000
0.000000	0.847793	0.0	0.0	0.0	0.860697	0.698362	0.000000	0	0.0	...	0.714008	0	0.000000	0.779887	0	0.693122	0.000000	0.0	0.783563	0.000000
0.747393	0.000000	0.0	0.0	0.0	0.794164	0.000000	0.752453	0	0.0	...	0.000000	0	0.834788	0.000000	0	0.704202	0.000000	0.0	0.879620	0.000000
0.000000	0.720058	0.0	0.0	0.0	0.746242	0.736189	0.000000	0	0.0	...	0.845264	0	0.815389	0.000000	0	0.782984	0.000000	0.0	0.828620	0.000000
0.000000	0.805688	0.0	0.0	0.0	0.800205	0.741403	0.000000	0	0.0	...	0.000000	0	0.699524	0.000000	0	0.758318	0.000000	0.0	0.768539	0.000000

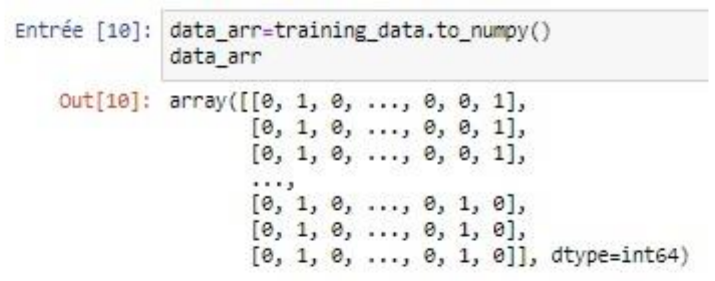
Figure 23 Example of UBRecordlink data that we used.

3.2 Splitting data

In this stage, an agent that has been trained on the source dataset is transferred into another agent that has been trained on a target dataset that is related to the source dataset (test data). To do that we make a file for the training data and the other for the testing data.

3.3 Encoding data to 0 and 1

We did this step to help us to know whether the item is existing or not, and to facilitate the creation of random bit vectors.



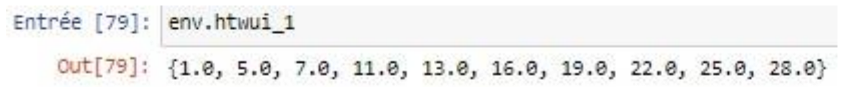
```
Entrée [10]: data_arr=training_data.to_numpy()
             data_arr

Out[10]: array([[0, 1, 0, ..., 0, 0, 1],
                [0, 1, 0, ..., 0, 0, 1],
                [0, 1, 0, ..., 0, 0, 1],
                ...,
                [0, 1, 0, ..., 0, 1, 0],
                [0, 1, 0, ..., 0, 1, 0],
                [0, 1, 0, ..., 0, 1, 0]], dtype=int64)
```

Figure 24: Encoding data step

3.4 Extract the items

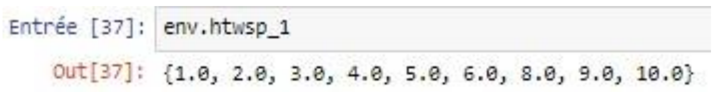
We notice that the cumulative reward was decreasing throughout the number of episodes, when we did the training with ubrecordlink data which has a lot of items, so the huge number of combined items used in the extraction of frequent itemsets and high utility itemsets is a crucial issue.

We decide to use HTWSP (high transaction weighted support pattern): {pro(in , TD) >= min_sup }and HTWUI (high transaction weighted utility itemsets): {u(in , TD) >= min_util } where we prune the search space and avoid database scan time by ignoring the non weighted items to have a smaller number of items which are high weighted, consequently we will decrease the length of the generated bit vector.



```
Entrée [79]: env.htwui_1

Out[79]: {1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 22.0, 25.0, 28.0}
```

Figure 25 The list of htwui from 29 items of UBRecordlink data



```
Entrée [37]: env.htwsp_1

Out[37]: {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 8.0, 9.0, 10.0}
```

Figure 26 The list of htwsp from 11 items of UBSkin data

So, this is the final step of preprocessing data, now we have all we need to move to the next step creating our model.

4. Model creation:

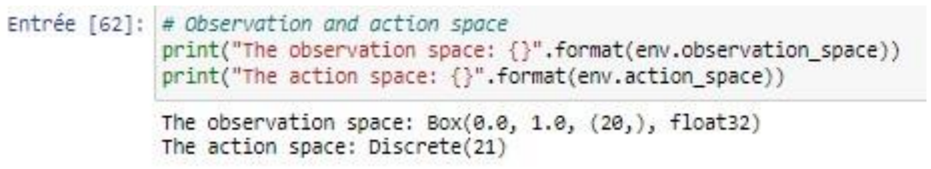
We go through the different steps to create our feature set extraction model:

4.1 Create the Environment

We create a specific environment using openAI gym, because we don't want to use an existing environment in our problem we have to implement gym.Env and redefine the functions:

- Render(): we use this function to visualize our results.
- Reset(): which resets the environment to its initial state and returns the observation to the initial state.
- Step(): this function takes an action as input and applies it to the environment, which causes the environment to transition to a new state. The step function returns four variables:

  - Observation: The observation of the state of the environment.
  - Reward: the reward you can get from the environment after performing the action given as input to the step function.
  - Done: Indicates if the episode has been completed. If true, you may need to end the simulation or reset the environment to restart the episode.
  - Info: This provides additional information depending on the environment.



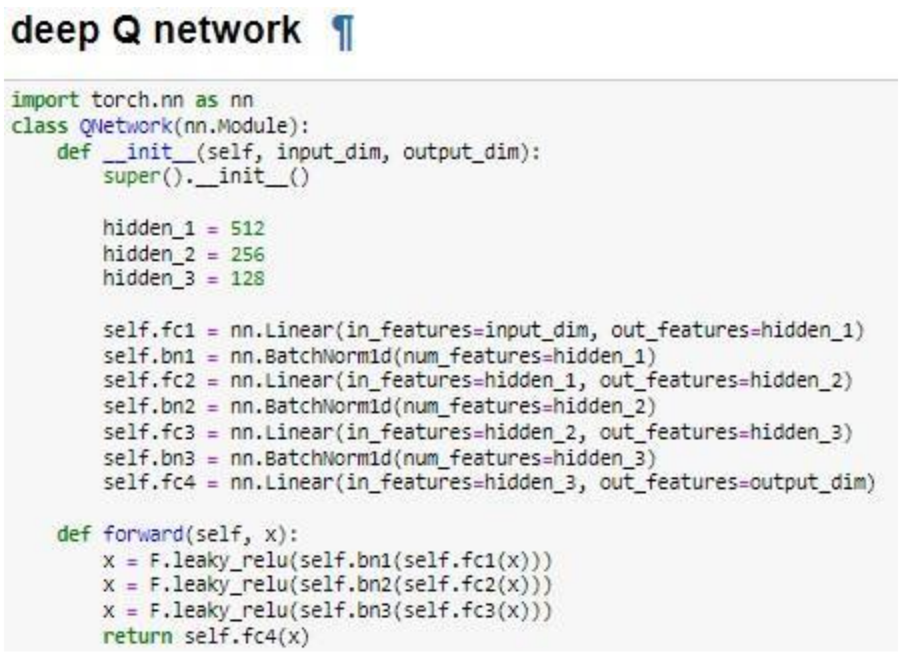
```
Entrée [62]: # Observation and action space
             print("The observation space: {}".format(env.observation_space))
             print("The action space: {}".format(env.action_space))

             The observation space: Box(0.0, 1.0, (20,), float32)
             The action space: Discrete(21)
```

Figure 27 observation and action space of our environment with UBRecordlink data

In our case, the observation_space was a box(0.0, 1.0, (18,), float32) and the action_space was discrete(19) as shown in figure 05. Box and Discrete are both types of data structures that come from the gym.space base class is provided by the Gym to describe legitimate values of observations and actions for environments.

4.2 Create the Deep Q Network

In this step, we construct our deep network using nn.module base class, the architecture is shown in Figure 32.

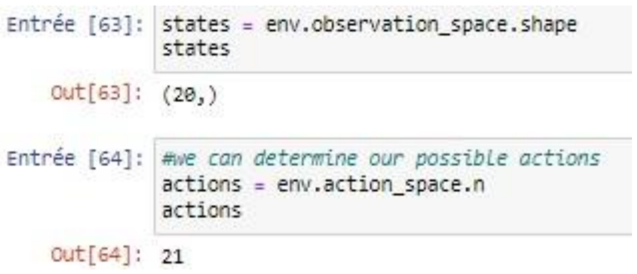


deep Q network ¶

```
import torch.nn as nn
class QNetwork(nn.Module):
    def __init__(self, input_dim, output_dim):
        super().__init__()

        hidden_1 = 512
        hidden_2 = 256
        hidden_3 = 128

        self.fc1 = nn.Linear(in_features=input_dim, out_features=hidden_1)
        self.bn1 = nn.BatchNorm1d(num_features=hidden_1)
        self.fc2 = nn.Linear(in_features=hidden_1, out_features=hidden_2)
        self.bn2 = nn.BatchNorm1d(num_features=hidden_2)
        self.fc3 = nn.Linear(in_features=hidden_2, out_features=hidden_3)
        self.bn3 = nn.BatchNorm1d(num_features=hidden_3)
        self.fc4 = nn.Linear(in_features=hidden_3, out_features=output_dim)

    def forward(self, x):
        x = F.leaky_relu(self.bn1(self.fc1(x)))
        x = F.leaky_relu(self.bn2(self.fc2(x)))
        x = F.leaky_relu(self.bn3(self.fc3(x)))
        return self.fc4(x)
```

Figure 28 Our DQN

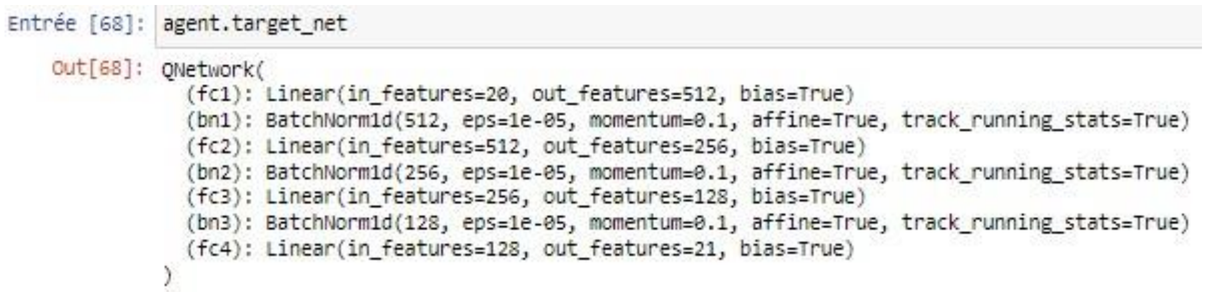
We had 18 inputs which is the number of states and the output layer produces an approximate quality of the action to change the item inclusion.



```
Entrée [63]: states = env.observation_space.shape
             states

   Out[63]: (20,)

Entrée [64]: #we can determine our possible actions
             actions = env.action_space.n
             actions

   Out[64]: 21
```

Figure 29 Our state and actions for the ubrecordlink data.

Our network consists of a fully connected layer and batch normalization layer and an activation layer where we used a leaky relu, all of these are collected in a block where we have 3 blocks.



```
Entrée [68]: agent.target_net

   Out[68]: QNetwork(
              (fc1): Linear(in_features=20, out_features=512, bias=True)
              (bn1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (fc2): Linear(in_features=512, out_features=256, bias=True)
              (bn2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (fc3): Linear(in_features=256, out_features=128, bias=True)
              (bn3): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (fc4): Linear(in_features=128, out_features=21, bias=True)
            )
```

Figure 30 Our model state after training the agent on UBRecordlink data.

4.3 Create the Agent

Our agent decides what action to take based on the q learning algorithm which determines the optimum action in the current situation and gets maximum rewards.

In reinforcement learning, gaining a reward from the environment is important, as is optimizing the agent to maximize the reward that is expected in the future. To do this, the agent will keep some data affected by the rewards he has received in the past and use it to make a better decision.

So, we define the agent with the constructor and six functions which are:

set network function where we set the deep network.

set optimizer function: In recent years, Adam has likely been the most widely utilized optimizer in Deep Reinforcement Learning research. that's why we used the Adam optimizer which is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update iterative network weights based on training data.

select action function based on q learning.

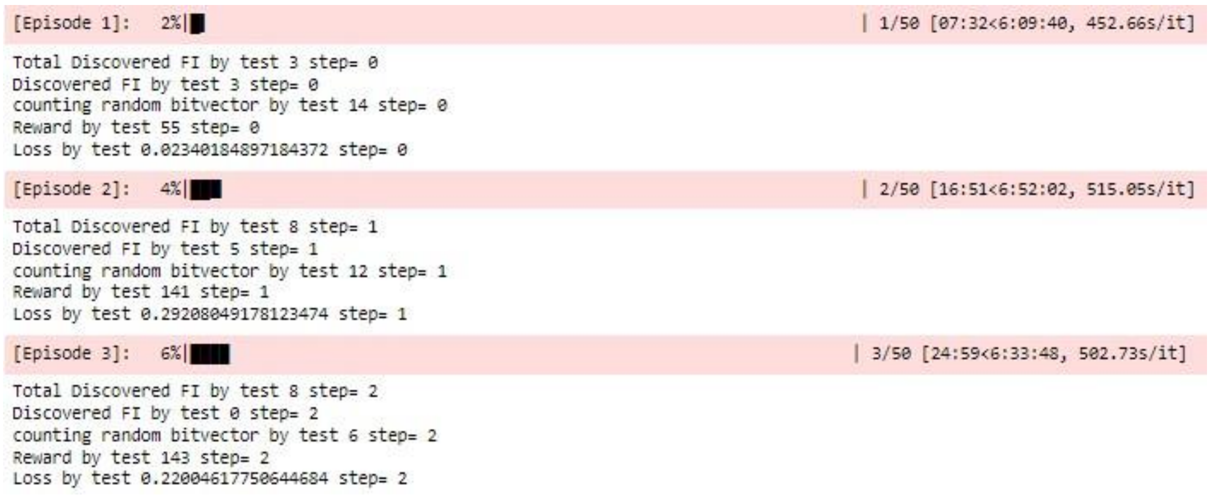
optimize agent function where we compute the loss and use our optimizer to minimize the loss and by minimizing the loss, we can find the optimal parameters that give the best performance of the model.

update agent function: get model function which returns policy network.

We have also created a replay memory with a capacity of 10000 where the agent stores his own experience.

4.4 Training step

In this step, we set the number of 50 episodes each episode consists of 5 steps and we let the agent train on the training data like is shown in the figure below.



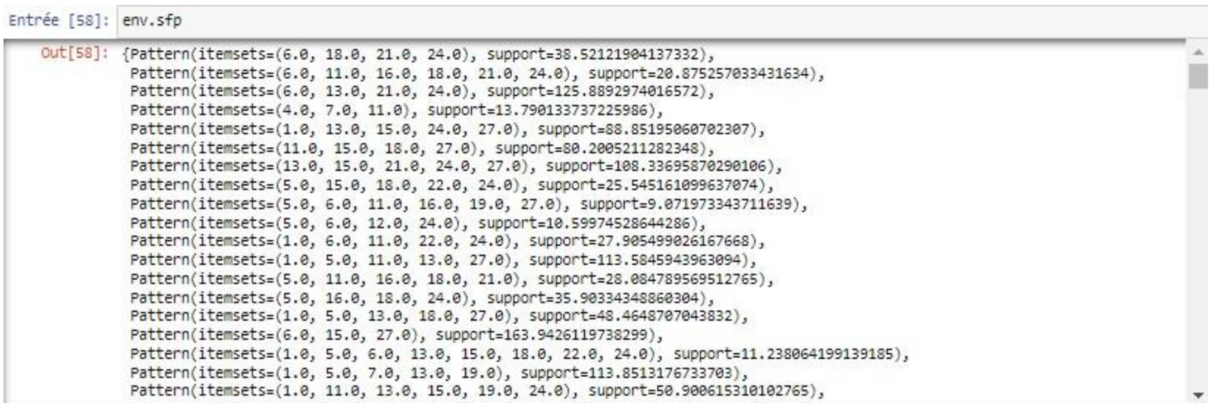
```
[Episode 1]:   2%|█                                                    | 1/50 [07:32<6:09:40, 452.66s/it]
Total Discovered FI by test 3 step= 0
Discovered FI by test 3 step= 0
counting random bitvector by test 14 step= 0
Reward by test 55 step= 0
Loss by test 0.02340184897184372 step= 0
[Episode 2]:   4%|██                                                   | 2/50 [16:51<6:52:02, 515.05s/it]
Total Discovered FI by test 8 step= 1
Discovered FI by test 5 step= 1
counting random bitvector by test 12 step= 1
Reward by test 141 step= 1
Loss by test 0.29208049178123474 step= 1
[Episode 3]:   6%|███                                                 | 3/50 [24:59<6:33:48, 502.73s/it]
Total Discovered FI by test 8 step= 2
Discovered FI by test 0 step= 2
counting random bitvector by test 6 step= 2
Reward by test 143 step= 2
Loss by test 0.22004617750644684 step= 2
```

Figure 31 Training step

4.5 The resulting file (itemsets with their support OR utility)

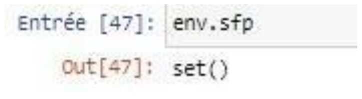
Finally, the extracted itemsets from the training data are saved with their support or utility in a named tuple and then saved in a file like it is shown in these figures.

1. **FI:**



```
Entrée [58]: env.sfp

    Out[58]: {Pattern(itemsets=(6.0, 18.0, 21.0, 24.0), support=38.52121904137332),
              Pattern(itemsets=(6.0, 11.0, 16.0, 18.0, 21.0, 24.0), support=20.875257033431634),
              Pattern(itemsets=(6.0, 13.0, 21.0, 24.0), support=125.8892974016572),
              Pattern(itemsets=(4.0, 7.0, 11.0), support=13.790133737225986),
              Pattern(itemsets=(1.0, 13.0, 15.0, 24.0, 27.0), support=88.85195060702307),
              Pattern(itemsets=(11.0, 15.0, 18.0, 27.0), support=80.2005211282348),
              Pattern(itemsets=(13.0, 15.0, 21.0, 24.0, 27.0), support=108.33695870290106),
              Pattern(itemsets=(5.0, 15.0, 18.0, 22.0, 24.0), support=25.545161099637074),
              Pattern(itemsets=(5.0, 6.0, 11.0, 16.0, 19.0, 27.0), support=9.071973343711639),
              Pattern(itemsets=(5.0, 6.0, 12.0, 24.0), support=10.5997452864286),
              Pattern(itemsets=(1.0, 6.0, 11.0, 22.0, 24.0), support=27.905499026167668),
              Pattern(itemsets=(1.0, 5.0, 11.0, 13.0, 27.0), support=113.5845943963094),
              Pattern(itemsets=(5.0, 11.0, 16.0, 18.0, 21.0), support=28.084789569512765),
              Pattern(itemsets=(5.0, 16.0, 18.0, 24.0), support=35.9033434886304),
              Pattern(itemsets=(1.0, 5.0, 13.0, 18.0, 27.0), support=48.4648707043832),
              Pattern(itemsets=(6.0, 15.0, 27.0), support=163.9426119738299),
              Pattern(itemsets=(1.0, 5.0, 6.0, 13.0, 15.0, 18.0, 22.0, 24.0), support=11.238064199139185),
              Pattern(itemsets=(1.0, 5.0, 7.0, 13.0, 19.0), support=113.8513176733703),
              Pattern(itemsets=(1.0, 11.0, 13.0, 15.0, 19.0, 24.0), support=50.900615310102765),
```

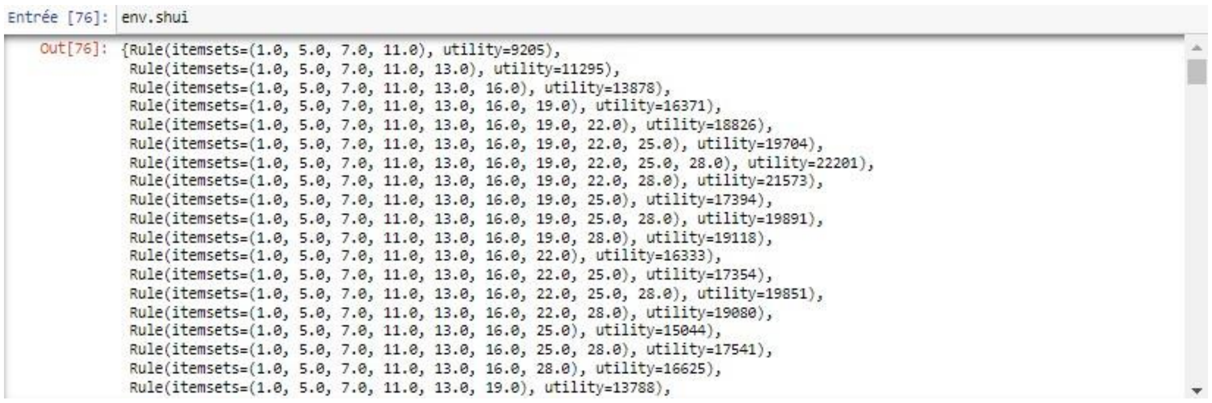
Figure 32 The tuple Pattern of UBRecordlink data with min support = 7.905050446999999 with expected support measure.



```
Entrée [47]: env.sfp

    Out[47]: set()
```

Figure 33 The tuple Pattern of the UBRecordlink data with min support = 7.905050446999999 with probabilistic support measure.

2. **HUI:**



```
Entrée [76]: env.shui
    Out[76]: {Rule(itemsets=(1.0, 5.0, 7.0, 11.0), utility=9205),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0), utility=11295),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0), utility=13878),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0), utility=16371),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 22.0), utility=18826),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 22.0, 25.0), utility=19704),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 22.0, 25.0, 28.0), utility=22201),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 22.0, 28.0), utility=21573),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 25.0), utility=17394),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 25.0, 28.0), utility=19891),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 19.0, 28.0), utility=19118),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 22.0), utility=16333),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 22.0, 25.0), utility=17354),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 22.0, 25.0, 28.0), utility=19851),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 22.0, 28.0), utility=19080),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 25.0), utility=15044),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 25.0, 28.0), utility=17541),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 16.0, 28.0), utility=16625),
              Rule(itemsets=(1.0, 5.0, 7.0, 11.0, 13.0, 19.0), utility=13788),
```

Figure 34 The tuple Rule of the UBRecordink data with min_util=9188.1.

5. Our proposed approach performance:

We can determine our RL model performance based on two important measures which are:

5.1 Efficiency

- **Execution time**

From the training and the testing step of our approach, we obtain these results presented in the table below.

UBSkin (200 transactions)
5:20:32

Table1 The time needed to find all FI with probabilistic support measure.

UBSkin data (11 items)	UBRecordLink (29 items)
00:12:02	+5:42:29

Table2 The time needed to find all FI (exp sup) with a different number of items.

UBRecordLink(40 transaction)	**UBRecordLink(200 transaction)**	**UBRecordLink(1000 transaction)**
+00:23:09	+1:47:50	+5:42:29

Table3 The time needed to find all FI (exp sup) with different number of transactions.

UBRecordLink(40 transaction)	**UBRecordLink(200 transaction)**	**UBRecordLink(1000 transaction)**
00:43:27	8:37:14	00:40:52

Table4 The time needed to find all HUI with different number of transactions.

- So, we can note that:
  - The time needed for each episode of our approach will be almost identical because the number of distinct items is fixed.
  - The time needed in a dataset with a little number of items will be shorter than a dataset with a lot of items.

- But it seems that the number of items (not the number of transactions) in the dataset is truly affecting the execution time so it needs more time and more episodes to extract all FI or HUI in a large number of items.

- **Memory consumption**

From the training and the testing step of our solution, we obtain these results presented in the table below.

	UBSkin		**UBRecordLink**	
Measure	**Expected sup**	**Probabilistic sup**	**Expected sup**	**Probabilistic sup**
TRAIN	49.5	76.3	49.1	69.4

TEST	48.9	75.8	50.5	65.7

Table5 The percentage of memory needed in extracting FI.

	UBRecordLink
TRAIN	55.9
TEST	59.4

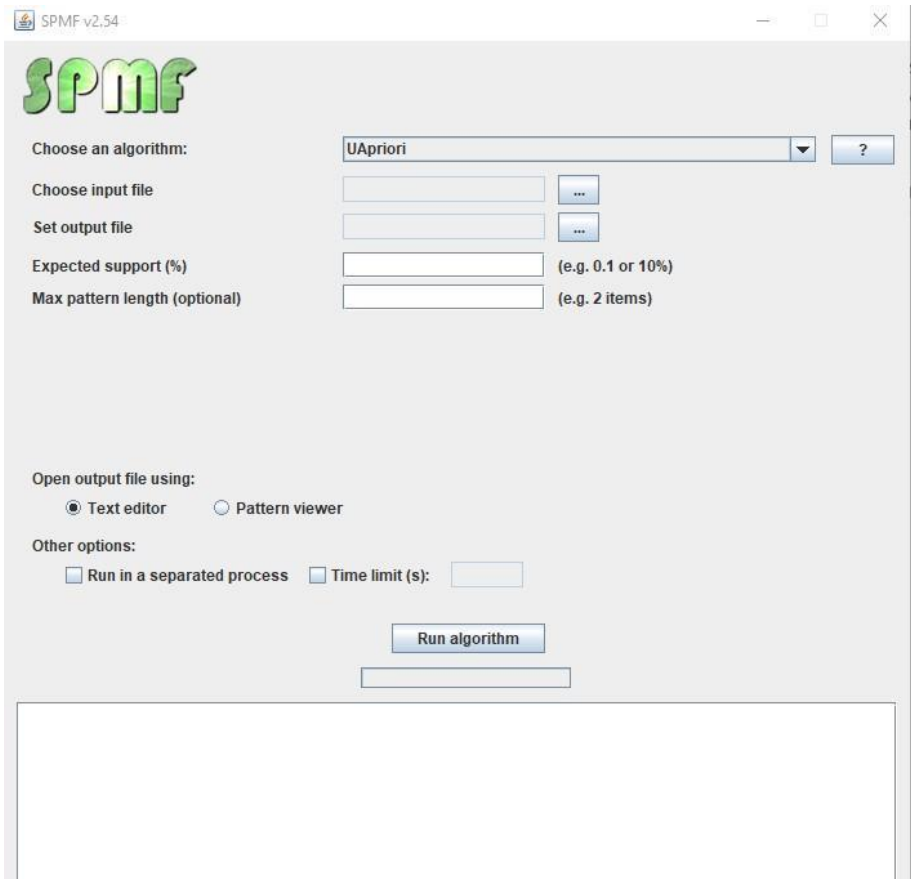
Table6 The percentage of memory needed in extracting HUI.

5.2 Quality of solutions

In this section, we will judge the quality of our solution to tell you about its advantages and disadvantages.

- So, our approach is an approximate approach because it can't find all the frequent items (or HUI) in an exact number of episodes.

  To verify the efficacity of our approach we must between the obtained result and the obtained results from UApriori And FHM (because they are an exact algorithms) in SPMF [37] which is an open source data mining library that offers a variety of data mining algorithm implementations (algorithms for extraction of FI with just expected support measure and for extraction of HUI). We downloaded the software and did some tests which are written in the tables below.


SPMF v2.54

Choose an algorithm: UApriori

Choose input file

Set output file

Expected support (%) (e.g. 0.1 or 10%)

Max pattern length (optional) (e.g. 2 items)

Open output file using:

[✓ Text editor] Pattern viewer

Other options:

Run in a separated process   Time limit (s):

Run algorithm

Figure 35 Spmf software's interface.

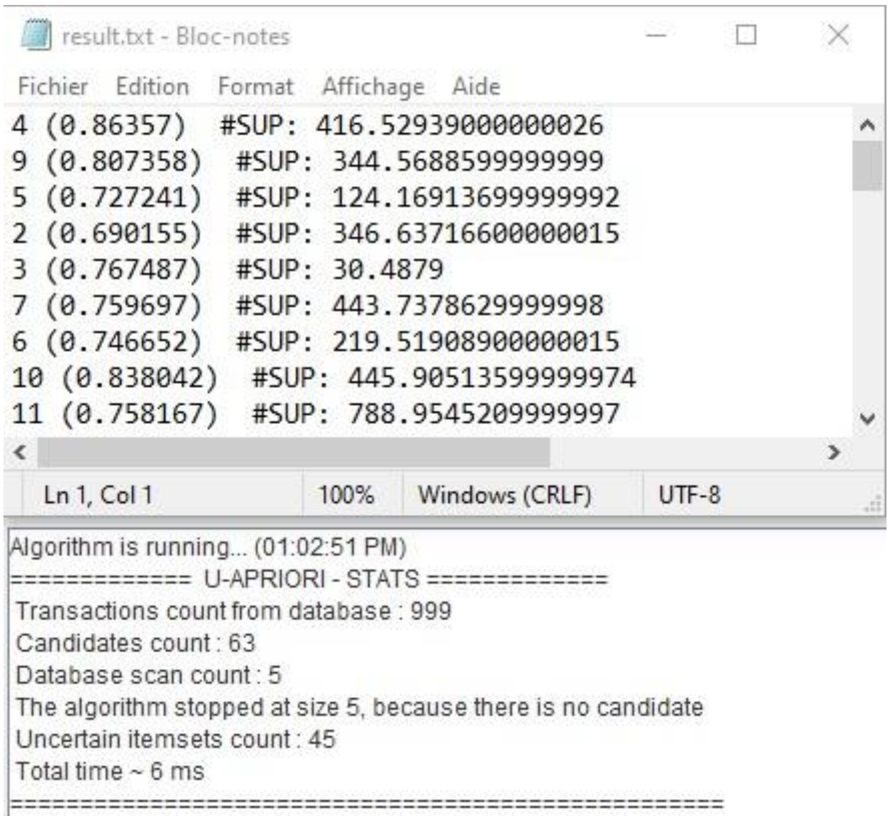
	Our approach	**UApriorialgorithm**
UBSkin	45	45
UBRecordLink	1155	6937

Table7 Comparison table between the result of the UApriori algorithm and our approach based on number of extracted FP from uncertain data.

	Our approach	**FHM algorithm**
UBRecordlink	867	867

Table8 Comparison table between the result of the FHM algorithm and our approach based on number of extracted HUI from uncertain data.

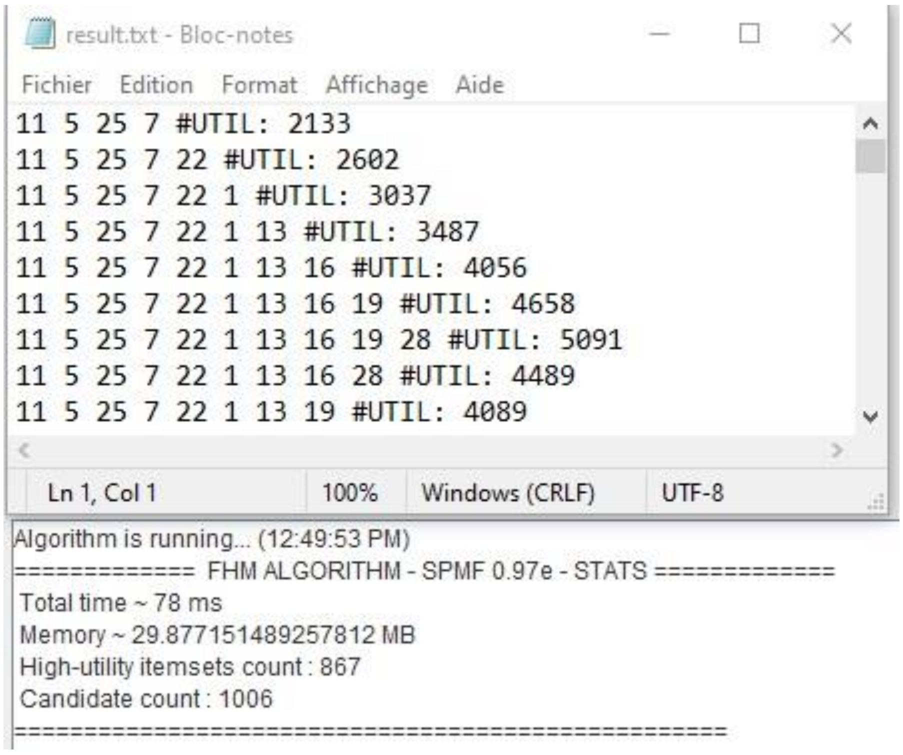
From the comparison between the result of the spmf software and our approach's result in the previous figure, we notice that there are no false positives (infrequent itemsets are considered frequent itemsets). But sometimes we can find false negatives (frequent itemsets not found).



```
4 (0.86357) #SUP: 416.52939000000026
9 (0.807358)  #SUP: 344.5688599999999
5 (0.727241)  #SUP: 124.16913699999992
2 (0.690155)  #SUP: 346.63716600000015
3 (0.767487)  #SUP: 30.4879
7 (0.759697)  #SUP: 443.7378629999998
6 (0.746652)  #SUP: 219.51908900000015
10 (0.838042)  #SUP: 445.90513599999974
11 (0.758167)  #SUP: 788.9545209999997
```

```
Algorithm is running... (01:02:51 PM)
============= U-APRIORI - STATS =============
 Transactions count from database : 999
 Candidates count : 63
 Database scan count : 5
 The algorithm stopped at size 5, because there is no candidate
 Uncertain itemsets count : 45
 Total time ~ 6 ms
===================================================
```

Figure 36 Example of Extracted FI from the small data with spmf software.



```
11 5 25 7 #UTIL: 2133
11 5 25 7 22 #UTIL: 2602
11 5 25 7 22 1 #UTIL: 3037
11 5 25 7 22 1 13 #UTIL: 3487
11 5 25 7 22 1 13 16 #UTIL: 4056
11 5 25 7 22 1 13 16 19 #UTIL: 4658
11 5 25 7 22 1 13 16 19 28 #UTIL: 5091
11 5 25 7 22 1 13 16 28 #UTIL: 4489
11 5 25 7 22 1 13 19 #UTIL: 4089
```

```
Algorithm is running... (12:49:53 PM)
============= FHM ALGORITHM - SPMF 0.97e - STATS =============
 Total time ~ 78 ms
 Memory ~ 29.877151489257812 MB
 High-utility itemsets count : 867
 Candidate count : 1006
===================================================
```

Figure 37 Example of Extracted HUI from the UBRecordLink data with spmf software.

- Consequently, we define the itemset mining effectiveness of our approach as the total number of episodes needed to extract all the frequent patterns. From the graphics below, we conclude that we need more episodes when we have a lot of items to extract all the FIs or HUIs.

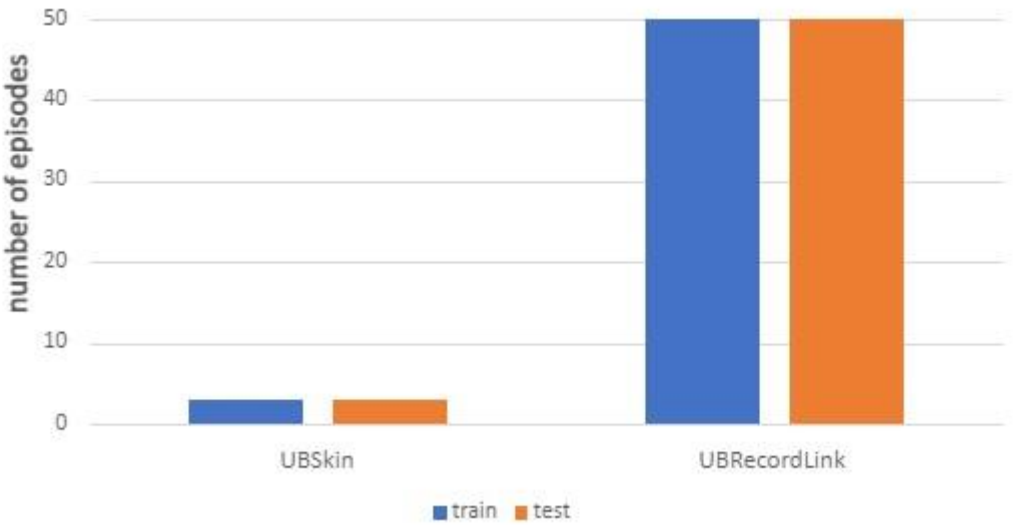


Figure 38 The number of episodes needed to find all FI.

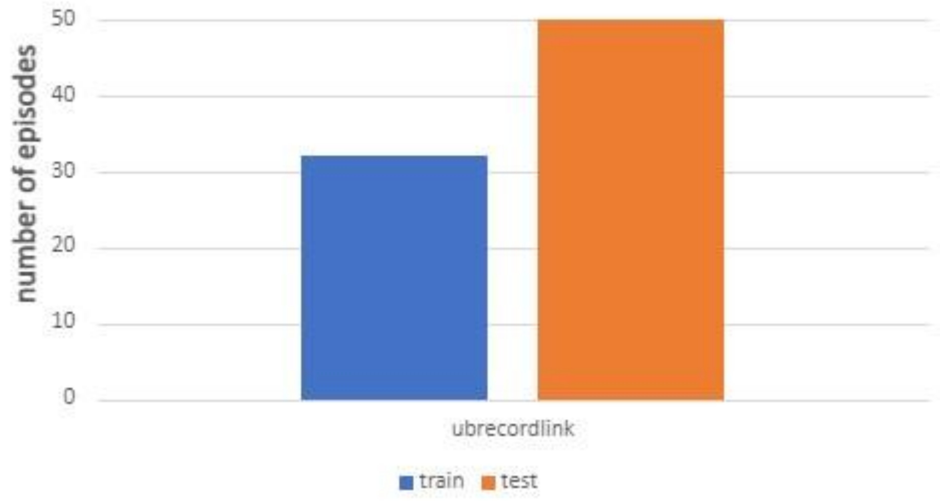


Figure 39 The number of episodes needed to find all HUI.

- A reinforcement learning algorithm can be evaluated based on the quality of the policy it discovers or the amount of reward it receives while behaving and learning.
  If an agent is required to learn while deployed, it may never reach the point where it is no longer required to explore, thus the agent needs to maximize the reward it receives while learning.
  The performance of our reinforcement learning algorithm is shown by the cumulative reward (the sum of all rewards received so far) as a function of the number of episodes in the figures below.

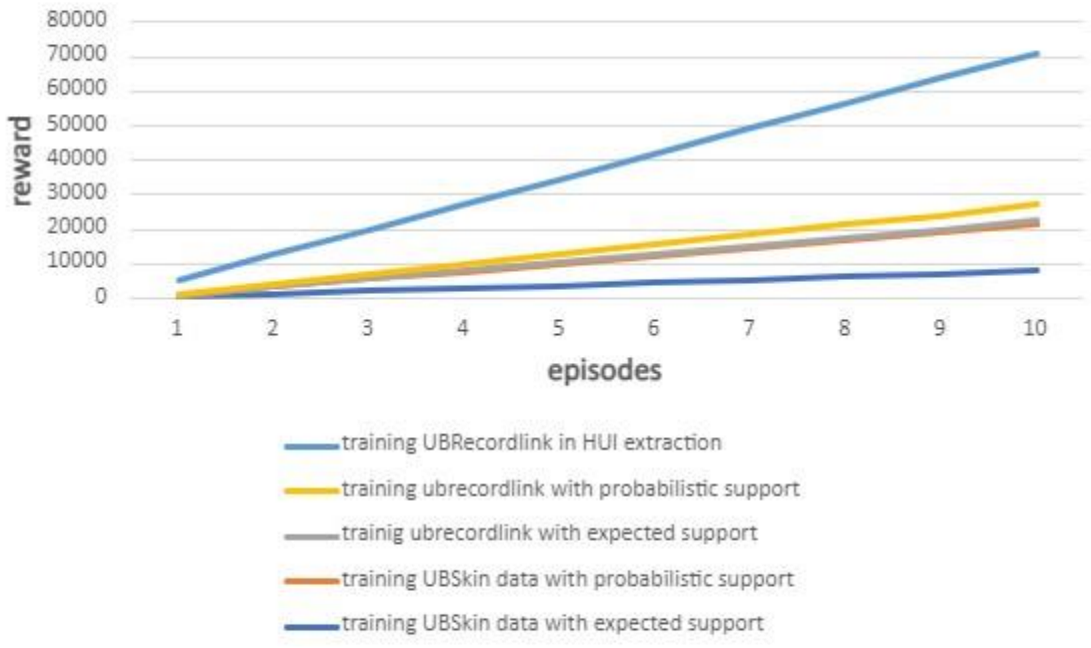


Figure 40 The cumulative reward as a function of the number of the first 10 episodes.

The performance of the algorithm is calculated by dividing the final reward by the total number of episodes. So, we can say that our approach is performance due to the increase of the reward.

- **Testing the SARSA algorithm:**

We also tested the SARSA algorithm where we update the q value by choosing a random q value of the next state, when we do that, we have seen that the algorithm chooses itemsets without relying on the best next action and without considering the high value of the reward.

For that, there is no guarantee to find all the FIs or HUIs and the reward will decrease like it is shown in the figure below.

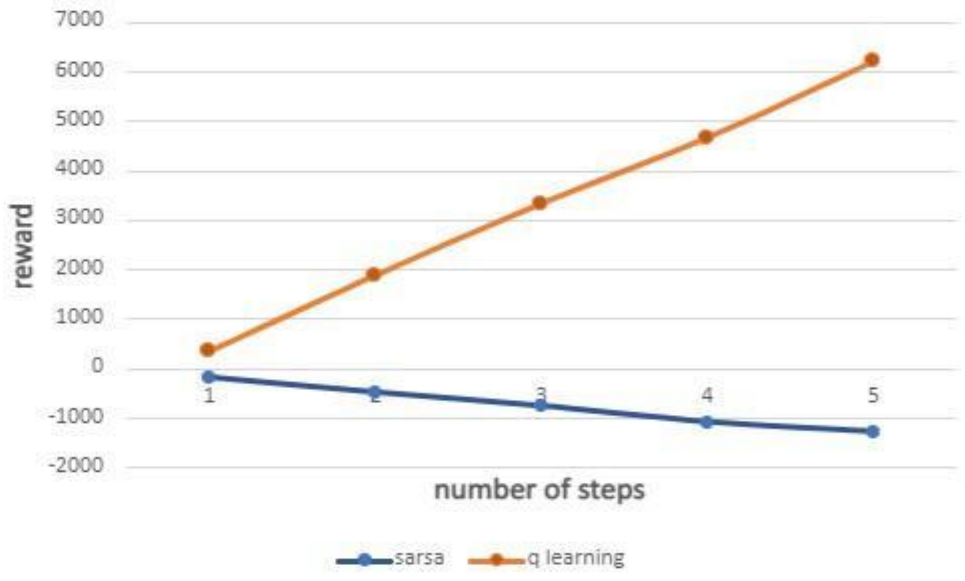


Figure 41 The difference between the Q-Learning and the SARSA algorithm in the cumulative reward as a function of the number of the first 5 episodes in FI**.**

6. Conclusion

We have presented the hardware and software environments, and the libraries used to implement our technique, and we have performed tests to evaluate the quality of our approach's outcomes in this chapter.

General Conclusion and Perspective

1. General Conclusion

Data processing is a rapidly expanding course of research that is becoming increasingly relevant, We have seen the extraction of frequent item sets, which is a method of data processing which always remains one of the most important subjects as a result of the expanding domains and uses for it, the extraction of patterns from uncertain data continues to be a prominent topic for scholars. While the scientific community produces various tools and processing methods.

In the course of our work to comprehend the challenge of extracting frequent itemsets and high utility itemsets from uncertain data, In the first chapter we've seen many approaches for extracting frequent itemsets and high utility itemsets from certain and uncertain data, and in the second chapter, we've researched and studied the various structures of machine learning and deep learning.

Then we discussed our deep learning-based solution “U-PMRL” and the training of an artificial agent that can extract frequent patterns from a database with the expected support and probabilistic support and how to extract a high utility items from uncertain data .Finally, we finished our work with the fourth chapter, test and experimentation, which allowed us to put our idea into practice and test it.

In this research, we introduced “U-PMRL”, a unified RL structure that allows for the extraction of different types of itemsets simply by modifying the reward definition. Through studies on the HUI, FI extractions, the general efficacy of U-PMRL is demonstrated.

This effort has allowed us to enhance our knowledge and better understand the concept of frequent item sets and high utility extraction in uncertain data and to discover and learn new techniques and many approaches concerning the deep learning.

But the biggest challenge we faced in validating and testing our approach was the absence of powerful equipment that allowed for additional experiments.

2. Perspective

This study is not a perfect model because it was created by a human, thus we are open to all criticism and are prepared to hear any recommendations or comments that can help make it better.

There are numerous perspectives on this exploratory study, and we will mention a few of them. For example To obtain the results and conduct comparisons use another computation measure, or try to use double DQN where we use two DQN one for selecting action and other for evaluate the actions or think about using transfer agent which generates a trained agent.

References

[1] X. Zhu, H. Deng, and Z. Chen, "A brief review on frequent pattern mining," *2011 3rd Int. Work. Intell. Syst. Appl. ISA 2011 - Proc.*, 2011, doi: 10.1109/ISA.2011.5873451.

[2] S. Nasreen, M. A. Azam, K. Shehzad, U. Naeem, and M. A. Ghazanfar, "Frequent pattern mining algorithms for finding associated frequent patterns for data streams: A survey," *ProcediaComput. Sci.*, vol. 37, pp. 109–116, 2014, doi: 10.1016/j.procs.2014.08.019.

[3] C. H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining: a literature review," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2603–2621, (2019), doi: 10.1007/s10462-018-9629-z.

[4] C. C. Aggarwal and J. Han, *Frequent pattern mining*, vol. 9783319078212. 2014.

[5] S. Agarwal, "Data mining: Data mining concepts and techniques," *Proc. - 2013 Int. Conf. Mach. Intell. Res. Adv. ICMIRA 2013*, pp. 203–207, 2014, doi: 10.1109/ICMIRA. (2013).45.

[6] M. B. Nichol *et al.*, "Quality of Anticoagulation Monitoring in Nonvalvular Atrial Fibrillation Patients : Comparison of Anticoagulation Clinic versus Usual Care," vol. 42, (2008), doi: 10.1345/aph.1K157.

[7] S. Yu and J. Watson, "Effective Algorithm for and Association," pp. 175–186.

[8] Y. Liu, W. Liao, and A. Choudhary, "A Two-Phase Algorithm for Fast Discovery of," pp. 689–695, 2005.

[9] S. R. Londhe, R. A. Mahajan, and B. J. Bhoyar, "Overview on Methods for Mining High Utility Itemset from Transactional Database," vol. 1, no. 4, pp. 15–19, 2013.

[10] C. F. Ahmed, S. K. Tanbeer, and B. Jeong, "Efficient Mining of High Utility Patterns over Data Streams with a Sliding Window Method," pp. 99–113, 2010.

[11] C. F. Ahmed, S. K. Tanbeer, B. Jeong, and Y. Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases," vol. 21, no. 12, pp. 1708–1721, 2009.

[12] B.-E. Shie, H.-F. Hsiao, and V. S. Tseng, "Efficient algorithms for discovering high utility user behavior patterns in mobile commerce environments," .

[13] Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. Data Knowl. Eng.59(3), 603–626, 2006

[14] Y. Liu, W. Liao, and A. Choudhary, "A Two-Phase Algorithm for Fast Discovery of," pp. 689–695, 2005.

[15] C. F. Ahmed, S. K. Tanbeer, B. Jeong, and Y. Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases," vol. 21, no. 12, pp. 1708–1721, 2008.

[16] Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu., P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. 25(8), 1772–1786

[17] J. C. Lin, W. Gan, T. Hong, V. S. Tseng, and J. C. Lin, “Efficiently mining uncertain high-utility itemsets,” *Soft Comput.*, 2016, doi: 10.1007/s00500-016-2159-1, 2005.

[18] U. Yun, H. Ryang, and K. Ho, “Expert Systems with Applications High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates,” *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3861–3878, 2014, doi: 10.1016/j.eswa.2013.11.038. 2007.

[19] Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.W., Tseng, V.S.: EFIM: a highly efficient algorithm for high-utility itemset mining. In: Proceedings of the 14th Mexican International Conference Artificial Intelligence, pp. 530–546. Springer,2015.

[20] C. Chui, B. Kao, and E. Hung, “Mining Frequent Itemsets from Uncertain Data,” pp. 47–58, 2007.

[21] Y Tong, L Chen, Y Cheng, PS Yu, "Mining frequent itemsets over uncertain databases," Proceedings of the VLDB Endowment, vol. 5, no. 11, pp. 1650-1661, 2012.

[22] J. C. Lin, W. Gan, P. Fournier-viger, T. Hong, and V. S. Tseng, “Knowledge-Based Systems Efficient algorithms for mining high-utility itemsets in uncertain databases,” 2016, doi: 10.1016/j.knosys.2015.12.019.

[23] Y. Lan, Y. Wang, Y. Wang, and D. Yu, “Mining High Utility Itemsets over Uncertain Databases,” pp. 2–5, 2015, doi: 10.1109/CyberC.2015.76.

[24] “Simple Reinforcement Learning: Q-learning | by Andre Violante | Towards Data Science.” https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56 , Oct. 25, 2022.

[25] W. Zhang and T. G Diettench, “A Reinforcemen t Learnin g Approac h t o Job-shop Scheduling,” *14th Int. Jt Conf. Artif. Intell*, pp. 1114–1120, 1995.

[26] “SARSA Reinforcement Learning - GeeksforGeeks.” https://www.geeksforgeeks.org/sarsa-reinforcement-learning/ Oct. 25, 2022.

[27] Y. S. Park and S. Lek, *Artificial Neural Networks: Multilayer Perceptron for Ecological Modeling*, vol. 28. Elsevier, 2016.

[28] “Recurrent Neural Network (RNN) : de quoi s’agit-il ?” https://datascientest.com/recurrent-neural-network Oct. 25, 2022.

[29] Y. S. Park and S. Lek, *Artificial Neural Networks: Multilayer Perceptron for Ecological Modeling*, vol. 28. Elsevier, 2016.

[30] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial,” *Computer (Long. Beach. Calif).*, vol. 29, no. 3, pp. 31–44, 1996, doi: 10.1109/2.485891,

[31] G, Dreyfus ; J. M. Martinez ; M. Samuelides M.B .Gordon ; F. Badran ; S .Thiria ; L. Hérault, ‹‹ Réseaux de neurones : Méthodologie et applications ››, Edition EYROLLES, 2016.

[32] Turchetti, Claudio (2004), *Stochastic Models of Neural Networks*, Frontiers in artificial intelligence and applications: Knowledge-based intelligent engineering systems, vol. 102, IOS Press,ISBN9781586033880.

[33] M. Groundwater, N. Contamination, and U. Artificial, “الشبكات العصبية الاصطناعية عبـدالنور عبـدالنور عادل عادل. د. د Neural Networks Neural Networks الشبكات العصبية الاصطناعية جامعة الملك سعود جامعة الملك سعود قسم الهندسة الكهربائية قسم الهندسة الكهربائية,” pp. 1–15, 2022.

[34] JV. Metsis, I. Androutsopoulos and G. Paliouras 2006. Spam filtering with Naive Bayes – Which Naive Bayes? 3rd Conf. on Email and Anti-Spam (CEAS).

[35] K. Fujioka and K. Shirahama, “Generic Itemset Mining Based on Reinforcement Learning,” *IEEE Access*, vol. 10, pp. 5824–5841, 2022, doi: 10.1109/ACCESS.2022.3141806.

[36] “Big Data Mining Reinforcement Learning RL 1071 BDM.” https://slidetodoc.com/big-data-mining-reinforcement-learning-rl-1071-bdm/ Oct. 25, 2022.

[37] “SPMF: A Java Open-Source Data Mining Library.” https://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php ,2022.

[38] M. M. Z. E. Mohammed, E. Bashier, and M. Bashier, *Algorithms and Applications*, no. July. 2016.

[39] “An Introduction to Deep Reinforcement Learning | Medium.” https://thomassimonini.medium.com/an-introduction-to-deep-reinforcement-learning-17a565999c0c ,Oct. 25, 2022.

[40] “Naive Bayes Classifier pour la Classification en Machine Learning.” https://mrmint.fr/naive-bayes-classifier, Oct. 25, 2022.

[41] N. Goyal, “A Comparative Study of Different Frequent Pattern Mining Algorithm For Uncertain Data : A survey,” vol. 1, no. Iccca, pp. 183–187,2022.