Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Detecting Physiological Needs Using Deep Inverse Reinforcement Learning

Khaoula Hantous, Lilia Rejeb & Rahma Hellali

To cite this article: Khaoula Hantous, Lilia Rejeb & Rahma Hellali (2022) Detecting Physiological Needs Using Deep Inverse Reinforcement Learning, Applied Artificial Intelligence, 36:1, 2022340, DOI: 10.1080/08839514.2021.2022340

To link to this article: https://doi.org/10.1080/08839514.2021.2022340

Published online: 07 Jan 2022.

APPLIED ARTIFICIAL INTELLIGENCE
2022, VOL. 36, NO. 1, e2022340 (1656 pages)
https://doi.org/10.1080/08839514.2021.2022340

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

OPEN ACCESS Check for updates

Detecting Physiological Needs Using Deep Inverse Reinforcement Learning

Khaoula Hantous, Lilia Rejeb, and Rahma Hellali

Université de Tunis, Institut Supérieur de Gestion de Tunis, SMART-LAB Strategies for Modelling and ARtificial inTelligence Laboratory, Tunis, Tunisie

ABSTRACT

Smart health-care assistants are designed to improve the comfort of the patient where smart refers to the ability to imitate the human intelligence to facilitate his life without, or with limited, human intervention. As a part of this, we are proposing a new Intelligent Communication Assistant capable of detecting physiological needs by following a new efficient Inverse Reinforcement learning algorithm designed to be able to deal with new time-recorded states. The latter processes the patient's environment data, learns from the patient previous choices and becomes capable of suggesting the right action at the right time. In this paper, we took the case study of Locked-in Syndrome patients, studied their actual communication methods and tried to enhance the existing solutions by adding an intelligent layer. We showed that by using Deep Inverse Reinforcement Learning using Maximum Entropy, we can learn how to regress the reward amount of new states from the ambient environment recorded states. After that, we can suggest the highly rewarded need to the target patient. Also, we proposed a full architecture of the system by describing the pipeline of the information from the ambient environment to the different actors.

ARTICLE HISTORY
Received 12 May 2021
Revised 30 November 2021
Accepted 9 December 2021

Introduction

Locked-in syndrome (LIS) or pseudocoma is a disease that causes quadriplegia and anarthria with consciousness preservation (Smith and Delargy, 2005). This means that the damaged patients are still aware of their environments, and they retain their vertical eye movement and blinking. Nevertheless, almost all their voluntary muscles are completely paralyzed, which causes a complete inability to communicate verbally. There are three categories in this disease: classic LIS, incomplete LIS, and complete LIS (Bauer, Gerstenbrand, and Rumpl, 1979). The first is defined by quadriplegia and dysarthria (motor speech disorder) with the capability of vertical eye movement. The second keeps the same symptoms as the first and adds other voluntary eye movement. The third type is defined by the immobility and

CONTACT Khaoula Hantous khaoulahantous@gmail.com Smart-lab Strategies for Modelling and ARtificial inTelligence Laboratory, Université de Tunis, Institut Supérieur de Gestion de Tunis, Tunis, Tunisie

inability to communicate, with full consciousness (Smith and Delargy 2005). There are many common causes of this syndrome, e.g., poisoning, overdose, nerve cell damage, and Stroke.

For LIS patients, there is only one *Alternative Communication* method that is eye communication. Patients count on their caregiver to understand them. The latter relies on eye-control methods which are *eye blinking* and *eye gaze*. There are two critical steps in the eye-tracking process: eye detection that consists in localizing the eye position in the captured image, and gaze estimation (also called gaze detection) that consists in estimating where the user is looking (Tatler, Witzner Hansen, and Pelz, 2019). Gaze tracking techniques are integrated into Alternative and Augmentative Communication (AAC) systems. They permit the user to communicate with others without talking. That is why such systems in persons with speaking disabilities' life represent a vital need (Light et al., 2019). AAC research field is concerned with assisted communication for people that have a speech impairment. One of AAC research's main aims is to investigate the possibilities of improving communication skills for non-speakers through the use of communication aids (Newell, Langer, and Hickey, 1998). There are several AAC systems that consider eye-tracking techniques. For instance, FEMA (Chareonsuk et al., 2016), which is a software designed for Amyotrophic Lateral Sclerosis (ALS) patients, enables them to use the computer as ordinary people using face and eye movements. They proved that the accuracy rate of left-click, right-click, double-click, scroll down, left movement and right movement is more than 80%. proposed an AAC architecture that helps researchers develop an adaptive communication environment at a low cost (Loja et al., 2015).

Besides, there are some commercial tools that are designed for similar type of users, and some of them have a patent that gives them the authority conferring their license or title for a set period like Tobii that is a leading eye-tracking technology by gaining many patents (Patents, 2020). Eye Tribe Tracker[1] is an eye-tracking device that can determine where the user is looking using details derived from his eyes. It calculates the coordinate of the point that the user is focusing on. The user should calibrate his eyes by editing the options. He should be in front of the trackbox in a manner that the system can theoretically track him. Eye Tribe Tracker is designed to help people with severe motor disabilities. It allows them to write texts, send e-mails, participate in online chats, etc. Moreover, it helps them in web browsing and pdf reading. Pages scroll when he reads on the bottom part of the page. The mentioned functionalities are offered by Eyegaze,[2] EyeControl[3] and GazePoint.[4]

It should be noted that the majority of the previous solutions provide a very low eye fatigue because they retain a high precision in the detection of the movement of the pupil. They are user-friendly and easy to use. However, the

patient takes a long time to express what he needs. The process is hard and slow. Patients are obliged to write a whole sentence using their eyes to express what they need. In some cases, they should rely on cards of needs where the caregiver is obliged to present all of them to the patient until understanding what he wants. That's why we are proposing a smarter solution that consists in detecting the needs according to the environment and the behavior of the target patient.

Our system's intervention commune in facilitating the patient's life by analyzing and trying to learn from the history of his behavior. The contribution consists in adding an intelligent layer to the AAC systems that are actually used by the target patients. It records, analyzes, and predicts the need depending on the history, knowing that when the patient approves the proposed suggestion, the system records the time and the state of the environment. To do so, we studied the two sides of RL. On the first side, RL seeks to learn the optimal behavior based on experiences, and IRL seeks to best understand and represent the observed behavior by learning the corresponding reward function. IRL introduces a new way of learning policies by deriving expert's intentions, in contrast to directly learning policies, which can be redundant and have poor generalization ability.

To choose the appropriate method for this problem, we reffered to (Coronato et al., 2020) who proposed a useful guideline for the application of RL to the health-care problems. They provide some indications to the designer in order to help him in choosing the appropriate RL method. We concluded that we should adopt the Inverse Reinforcement Learning (IRL) paradigm because of the following five facts: (1) The problem involves a multi-step decision process. Patient's needs are expressed sequentially. There is no complete and accurate model of the environment. Patient's environment is difficult to present. (2) The actions and states can be presented as arrays. (3) We cannot let the agent interact with the real environment because we cannot propose all the possible needs at each time step, and let the patient answer through trial-and-error. Also, we cannot conclude his preference since he can change them depending on his mind or mood. (4) We are not able to model the reward function R for every possible state features. After studying the IRL variants, we decided to adopt the Deep Inverse Reinforcement Learning using Maximum Entropy since multiple reward functions can explain the patient's behavior, and Maximum Entropy variants are the best to represent the problem when there are multiple-reward functions that can explain the expert's behavior (Arora and Doshi, 2021).

This paper is organized as follows: Section 2 presents the notions of RL and IRL. Section 3 introduces our new approach of need detection. It consists in detailing the *Intelligent Communication Assistant* by explaining the architecture of the new AAC system with a full details about the front-end and back-end algorithms. Section 4 is dedicated to the logical assumptions that we

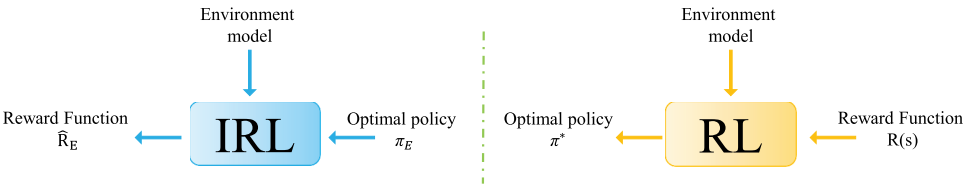


Figure 1. The two RL paradigms.

followed to simulate a data that presents the patients needs. Section 5 presents the experimentation and the obtained results that prove the effectiveness of the proposed approach. To conclude, a brief discussion is proposed in Section 6, in which we aim to present the open issues that we identified and some future works to be done.

Reinforcement Learning

Reinforcement Learning (RL) is a computational approach that consists in determining how to map among situations by enunciating actions that maximize the numerical reward signal. The learner does not know which actions to take, but instead, he must discover which one will yield the best situation (Sutton and Barto, 1998; Sutton and Barto, 2018). This section presents the background of RL because it is essential to have a good understanding of RL paradigm, to be able to understand IRL paradigm and the other variants. As mentioned in Figure 1, those two paradigm use the same terminologies that are as follows: Environment, Policy and Reward. But, there is a difference between the input and the output of their algorithms.

Basic Notions

According to Sutton and Barto, the process of *sequential decision-making* contains two steps: The agent is provided with an initial observation of the environment, and he is required to choose some actions from the given set of possibilities (Sutton and Barto, 1998). Those steps incorporate six fundamental elements:

- *The environment* defines the world that the agent interacts with, and it has a primary loop. In each iteration, it produces a state and a reward for the agent to sense and process. Then, it receives an action from the agent and cycles back to produce another state again.
- The *agent* learns to achieve goals by interacting with the environment. In each iteration, the agent senses the state, then he chooses an action and receives the appropriate reward from the environment.

- The *state* (noted as s_t) represents a situation of the environment that occurs at the time-step t. It describes the shape of the environment based on which the agent is going to choose his actions.

There are three types of states: a *discrete state* like a tic-tac-toe game, a *high-dimensional state* like the pixels in a video game and a *continuous states* like the industrial controller with continuous values (temperature and pressure).

- The *Time-step* (noted as t) divides time into discrete steps. Each one of them determines a cycle in the environment–agent interaction.
- The *reward* (noted as R_t) is a scalar value, a floating-point number, returned by the environment when the agent selects an action. It represents the goal or a set of goals and depends on the reinforcement learning problem designer.
- The *action* (noted as a_t) is one of the possible choices of decisions that the agent can take in each time-step. It can be discrete or continuous. Discrete actions are limited with a set of possible actions like {Left, Right, Up, Down}.

The *Problem setup* is presented by the interaction between the described elements. The Interface is composed of an environment that exposes the state, and an agent who senses that state, and takes an action. The environment processes the action and produces two things: a reward and a new state. This cycle continues a certain number of times. The agent interacts with the environment over time. At each time step t, the agent receives a state s_t from a state space S, and selects an action a_t from an action space A, following a policy $\pi(a_t|s_t)$. The agent receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics, or model, for reward function $R(s, a)$ and, state transition probability $P(s_{t+1}|s_t, a_t)$, respectively. In an episodic problem, this process continues until the agent reaches a terminal state. After that, it restarts. The return is the discounted, accumulated reward multiplied with the discount factor $\gamma \in [0, 1]$ (Yuxi, 2018).

Reinforcement Learning (RL)

Reinforcement learning problems are formulated using the Markov property (Markov, 1954), which enunciates that the current state characterizes the state of the word and depends on the current observation, i.e., there is no need to back the whole history and consider all the previous states. A Markov Decision Process (MDP) is defined by the quintuple (S, A, R, P, γ) (Bellman, 1957). It is a discrete-time stochastic control process that assume that the effect of taking an action at a given state only depends on the present state-action pair and not on the previous states and actions.

The goal of reinforcement learning algorithms is to design the policy. A policy π is a function that specifies what action to take in each state ($S \rightarrow A$) toward reaching the objective that is maximizing the cumulative discounted reward (Equation 1). The goal of reinforcement learning algorithms is to design the set of state-actions that constitute the policy. As a result, their output is defined by the specific sample of trajectories: $s_0, a_0, r_0, s_1, a_1, r_1, ..etc.$

$$\max \sum_{t>=0} \gamma^t r_t \tag{1}$$

Thereby, the goal of the learner is to find the optimal policy π^* (Equation 2).

$$\pi^* = \arg\max_{\pi} \left[E[\sum_{t\geq 0} \gamma^t r_t | \pi \right] \tag{2}$$

The value function is the prediction of the expected, accumulative, discounted, future reward, measuring how good each state, or state-action pair, is (Yuxi, 2017). The state-value is obtained by calculating the expected return following the policy π in the state s. It is calculated using Equation 3.

$$v_\pi(s) = E[R_t | s_t = s] \quad with \quad R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3}$$

On the other side, there is a known class of algorithms useful for solving reinforcement learning problems: the class of *evolutionary algorithms*. According to the taxonomy made by (Ivanov and D'yakonov, 2019), this class of algorithms cannot be considered the fifth class because it does not utilize the RL structure like the other described classes of algorithms.

Inverse Reinforcement Learning (IRL)

The second RL paradigm is Inverse Reinforcement Learning (IRL) which is a learning approach capable of inferring the Reward Function of an agent, given its policy or observed behavior" (Arora and Doshi, 2018). Given the execution traces, IRL algorithms are capable of recovering the reward R i.e., given the policy π^* and the dynamic D.

$$D = \{\tau_1, \tau_2, ..\tau_n\} \tag{4a}$$

$$\tau = (s_0, a_0), (s_1, a_1), .., (s_j, a_j) \tag{4b}$$

IRL considers the problem of extracting a reward function from observed (nearly) optimal behavior of an expert acting in an environment (Abbeel and Ng, 2010).

IRL

IRL consists in modeling the preferences of another agent using its observed behavior, thereby avoiding a manual specification of its reward function (Arora and Doshi, 2018). IRL is very efficient because of its potential to use recorded-data in a task to build autonomous agents capable of modeling others by catering to the following needs:

- Learning from an *expert* by creating an agent using the expert's preferences. E.g, helicopter flight control by (Abbeel et al., 2007), an expert helicopter operators' sophisticated preferences were learned from recorded behavior data using IRL. This reward function was then used to teach a physical remotely controlled helicopter advanced maneuvers using RL.
- Learning from another *agent* to predict its behavior. E.g, route prediction for taxis (Ziebart et al., 2008b). The idea consists in training a model using the route preferences of 25 taxi drivers demonstrated in over 100 000 miles of collected data. They demonstrate the performance of the proposed model by inferring both decisions at the next intersection, the route to a known destination, and the destination given partially traveled route

To sum up, IRL supposes that the expert or the agent behaves according to an underlying policy π_E, which may not be known. If a policy is unknown, the learner observes the sequence of state-action (also called trajectories) built by the expert. Here, the reward function is unknown, but the learner usually needs some structure that helps in the learning (Arora and Doshi, 2018).

IRL problem is first formally studied by Russell in 1998 and described like follow: Given the MDP's tuple, IRL is capable of determining a set of possible reward functions R such that π is the optimal policy. Since the objective of IRL is to learn the unknown R^*, it is capable of handling the following possible cases:

(1) *Finite-state MDP with known optimal policy* that represents the easiest scenario defined by a finite-state MDP with a known and completely observed policy. For the ease of representation, the optimal policy is given by $\pi(s) = a_1$.
(2) *Infinite-state MDP with known optimal policy* that describes infinite state space problems in which the reward function is approximated using a linear combination of useful features (Ng and Russell, 2000).
(3) *Infinite-state MDP with unknown optimal policy* also called *Apprenticeship Learning with Sample trajectories*. It is designed to handle complex applications in which the desired trajectory is hard to describe e.g., the helicopter aerobatic maneuver trajectory that depends on the helicopter

dynamics (Abbeel, Coates, and Ng, 2010). The demonstrations provided by the expert can be used to extract the desired trajectory through apprenticeship learning.

This classification was made by (Zhifei and Meng Joo, 2012), and they proposed a detailed explanation about the objective function, input and output of each case. In our case, the patient's room and needs represents an *infinite-state MDP with known optimal policy* because we aim to simulate the optimal policies that describe the patient's behavior depending on the patient's room.

IRL Methods

IRL involves solving the MDP with the function hypothesized in current iteration and updating the parameters, constituting a search that terminates when the behavior derived from the current solution aligns with the observed behavior. According to Arora and Doshi (2021), the foundational methods for IRL are:

(1) *Max margin methods* which maximize the margin between the values of the observed behavior and the hypothesis i.e., they cover a solution that maximizes some margin. For example: minimizing the margin between the feature expectations of a policy computed by the learner and the empirically computed feature expectations from the expert's trajectory.
(2) *Max entropy methods* maximize the entropy of the distribution over behaviors. According to this principle, the distribution that maximizes the entropy makes minimal commitments beyond the constraints and is least wrong. We broadly categorize the methods that optimize entropy based on the distribution, whose entropy is being used, that is chosen by the method.
(3) *Bayesian learning methods* learn posterior over hypothesis space using Bayes rule. This means that the Bayesian IRL methods are based on how they model the observation likelihood.
(4) *Classification and Regression* learn a prediction model that imitates observed behavior. IRL may be formulated as a multi-class classification problem by viewing the state-action pairs in a trajectory as data-label pairs.

Maximum Entropy methods are useful when multiple-reward functions can explain the expert's behavior (Arora and Doshi, 2021). Consequently, we decided to adopt it because multiple reward functions can explain the patient's behavior.

Deep IRL Using MaxEnt

Inspired from the solution of (Ziebart et al., 2008a) who, first, applied the maximum entropy principle to solve IRL problems for cases where the reward function depends only on the current state. At this phase, we aim to present

the relationship between states, actions, rewards, environment features and neural network weights using the demonstration of Ziebart et al. Rewards are represented by a linear sum of weighted features as shown in Equation 5. $\Phi : S \rightarrow \mathbb{R}$ is the feature set, w^T is the weight vector, and R is the reward vector.

$$R(s,a) = w^T \Phi(s). \tag{5}$$

Note that the feature vector $\Phi(s)$ is a function of states only, and actions were not considered. To represent the couple of (state, action) considering the feature vector $\Phi(s,a)$, where Φ is a function of states and actions, the learner may consider reward functions as a linear combination of features, as shown in Equation 6 (You et al., 2019).

$$\begin{aligned} R(w;s,a) &= w_1\boldsymbol{\phi}_1(s,a) + w_2\boldsymbol{\phi}_2(s,a) + \ldots + w_k\boldsymbol{\phi}_k(s,a) \\ &= w^T\Phi(s,a) \end{aligned} \tag{6}$$

The use of non-parameterized features requires to design the features manually, which may be a difficult task since it may not always be possible to approximate a certain unknown reward functions having a complicated form. Hence, You et al. (2019) consider the use of parameterized features using the parameter vector θ.

$$R(w,\theta;s,a) = w^T\Phi(\theta;s,a) \tag{7}$$

Equation 7 tunes the vector θ besides of tuning only the weight vector w^T in order to maximize the likelihood. Consequently, we can calculate Equation 8 using NN backpropagations by following the gradient descent.

$$\frac{\partial R(w,\theta;s,a)}{\partial\theta} \tag{8}$$

We aim to train the model based on the history of the patient. In other words, from his optimal choices that represent the optimal policies. The expert E is the Intelligent Assistant who acts according to the dynamic D. The latter presents the recorded optimal policies of each episode. Equation 4 defines the formulation of this dynamic. Each trajectory τ is defined by the trajectory $(s_0,a_0),(s_1,a_1),..,(s_j,a_j)$ that describe each couple of (state, need).

Our goal is to design an algorithm capable of learning from those optimal trajectories in order to be able to suggest the most fittable needs in the new unknown situations, i.e., new ambient environment records. To learn the reward function, Chen et al. (2019) introduced the characteristic function that needs to be specified artificially. As a result, the reward function is defined by the Equation 9.

$$r_\theta(s) = \theta^T\boldsymbol{\phi}(s) \tag{9}$$

$\phi(s)$ is an artificial characteristic function. One solution is to use neural networks to represent the reward function. At this point, the reward function can be expressed like illustrated in Equation 10 where $f(s)$ is the characteristic function as shown in Figure. This representation is proposed by (Chen et al., 2019).

$$r(s) = \theta^T f(s) \tag{10}$$

A New Intelligent Communication Assistant

As explained before, the idea behind this research work is to propose a new *Intelligent Communication Assistant* able of detecting the needs of LIS patient. In this section, we will introduce the ambient environment that is necessary to feed the system. After that, we studied human needs from the psychology perspective and then extracted the patients needs. Then, we present the proposed algorithms for need detection. Finally, we present the full architecture of the *Intelligent Communication Assistant*.

Ambient Patient Environment

To enhance patient care, we suppose that the patient follows his healing process in a smart environment that employs ambient intelligence (AmI) technologies. For instance, (Kartakis et al., 2012) designed a smart patient room with a user interface development to assist both patients and medical staff in order to enhance Health-Care Delivery through Ambient Intelligence Applications. Importantly, AmI supports the pervasive diffusion of intelligence in the patient environment thanks to wireless technologies (e.g., Zigbee, blacktooth, RF, Wi-Fi, and intelligent sensors). Many other research works focused on enhancing health-care services through AmI applications. We took the problem from the data scientist's perspective. What caught our attention is data delivered from those intelligent environments and how we can use it to ameliorate the patient life. Smart patient rooms can record data related to the patient (e.g., recording the heart beating, blood pressure, pulse oximetry, etc.) and related to his environment (e.g., temperature, bed, tv, window, door, visitor detecting, etc.).In our work, we consider six features that describe the patient's room in order to catch the environment's state, as mentioned in Figure 2.

With respect to the notation of RL that we presented in Section 2, we consider that S is the set of the possible states that represents the dynamic environment where each state is defined by the six features that we defined in Table 1.

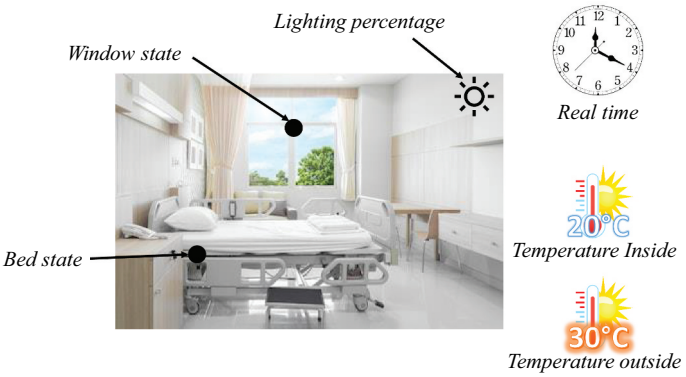


Figure 2. Abstracted illustration of the patient's ambient room.

Table 1. A brief description of the environment' recorded features.

Environment's feature	Description
Time	Saving the real time, at which the state is detected.
Room Temperature	Tracking the temperature inside the patient's room that must be between 18 and 20 degrees. This caliber is not affected by the change of seasons.
Exterior Temperature	Tracking the temperature of the exterior i.e., outdoor temperature.
Lighting percentage	Recording the percentage of the brightness inside the room. Values are in between of 0 and 100.
Window state	Picking the state of the window that can be either open or close. In the first case, values are between [0.1], and in the second case values are between [1.2].
Bed state	Picking the state of the bed that can be either low, intermediate or high. If values are between [0.1], the bed is low. If values are between [1.2], the bed is intermediate. Finally, If values are between [2.3], the bed is high.

Patient's Needs

Maslow is the first to introduce the concept of the hierarchy of needs, in which he suggests that normal people search to accomplish the most basic needs for survival before searching for the more complex ones. He focused on the problematic of studying behaviors by learning the facts that make people happy and the things they do to achieve that aim. The Hierarchy of Needs presents a motivational theory in psychology that groups human needs into the five following levels: *Physiological Needs, Security and Safety Needs, Social Needs, Esteem Needs*, and *Self-Actualization Needs* (Maslow, 1943).

A good deal of psychological research criticized Maslow's classification of needs. Some of them reported that needs doesn't really follow a hierarchy (Wahba and Bridwell, 1976). On the other hand, some other researchers were influenced by Maslow's contributions and considered them as a big shift in psychology. For instance, (Tay and Diener, 2011) put the hierarchy to test and discovered that the fulfillment of the needs was strongly correlated with happiness. We concluded that satisfying human needs is not an easy task because they differ from one person to another. On the first side, we have needs that depend on the studied

person, his background, center of interest, hobbies, relationships, etc. On the other side, we have LIS patients who only want to accomplish the basic survival needs, and to balance between those two sides, we can treat the basic survival needs that belong to the lowest level by focusing on the *physiological needs*.

Physiological needs aim to satisfy the body homeostasis requirements, e.g., adjusting the room temperature, opening or closing the window, maintaining the bed state, augmenting, or reducing the light. It treats Nutrition and Hygiene needs, e.g., making dinner, brushing teeth, shaving, or getting a shower. LIS patients cannot use their jawbones because they cannot move the muscle of their mouths. Thus, we excluded nutrition needs except water drinking. Hygiene needs are divided into two types: body hygiene and environmental hygiene. Body hygiene is related to the body state and time, but environment hygiene is directly related to the environment. After studying the disease and understanding the human needs process, we are aware that the satisfaction of needs is not an "all-or-none" phenomenon. We can not deal with all the patient's needs. This explains why we oriented ourselves to the needs related to the environment. Along with this, we will focus on the *body homeostasis needs* by designing the appropriate ambient patient environment.

As mentioned previously, this work will rely on the chosen six features (See Figure 2) by considering the following needs: *Open the window, Close the window, Change the bed position, Move, Drink, Set the temperature, Sleep* and *Adjust the light. Body homeostasis needs* will be detailed in the subsection 4.1, at which, a full explanation of the considered assumptions for need representation is presented. With respect to RL notation, A is the finite set of available actions that we defined previously.

Learning the Physiological Needs

At this phase, we aim to present Algorithm 1 that is able to take the set of time-recorded states S, set of action A, reward function R and the dynamic of the environment D as an input (see equation 4). After that, the neural network is trained depending on the content of D by assigning the room state observations' values to the input and the designed reward amount to the output.

Algorithm 1 Deep Inverse Reinforcement Learning to learn the patient's behavior

Input: $S, A, D, n_{episode}$

Output: $\hat{R}$ ▷ Estimation of the reward payoff for any new couples of (s_t, a_t)

1: Randomize the weights w to small random values ▷ Randomizing the neural network weights

2: $observation\_index \leftarrow 0$ ▷ The index of the recorded states

3: Initialize the input vector *InVect*
4: Initialize the output vector *OutVect*
5: **for** episode = 1 to $n_{episode}$ **do**
6: **for each** $(s_t, a_t) \in \tau_{episode}$ **do**
7: $observation\_index \leftarrow observation\_index + 1$
8: **for each** $a_i \in A$ **do**
9: **if** The patient accepts the suggested need **then**
10: $Payoff[i] \leftarrow 1$
11: **else**
12: $Payoff[i] \leftarrow -1$
13: $InVect[observation\_index] \leftarrow s_t$
14: $OutVect[observation\_index] \leftarrow Payoff$
15: **for** epoch = 1, Epochs **do** ▷ Epochs represents the total number of epochs
16: Apply the network input vector *InVect* to network
17: Calculate the output vector of the network using *OutVect*
18: Calculate the errors for each one of the outputs
19: Calculate the necessary updates for weights Δw to minimize this error
20: Add up the calculated weights' updates Δw to the accumulated total updates updates ΔW
21: Adjust the weights w of the network by the updates ΔW

To simplify the proposed pseudocode, we illustrated the process of the neural network in Figure 3.

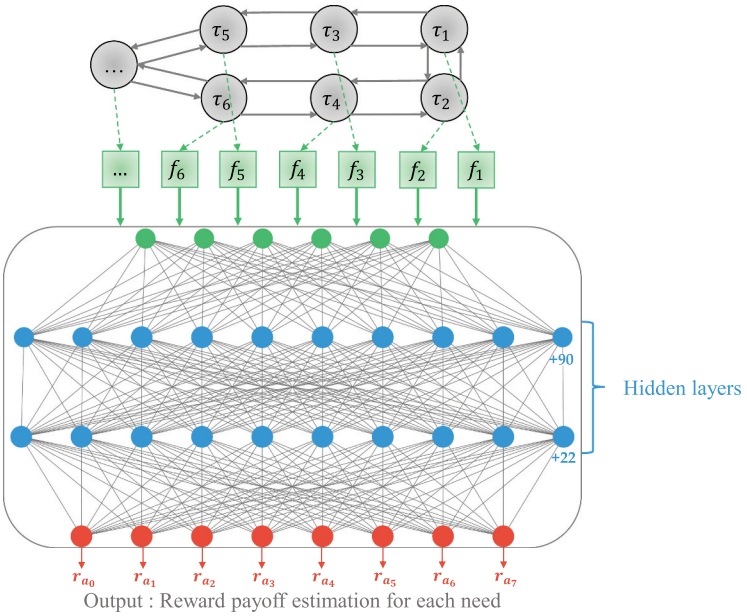


Figure 3. Reward function approximation based on deep natural network.

For each recorded trajectory τ, the algorithm associates a new *observation_index* and then fills the *InVect*[*observation_index*] with the environment feature values i.e., Time, Room Temperature, Window state, etc. and associates to the *OutVect*[*observation_index*] the corresponding reward amount for each action (See Figure 4). Finally, the Neural network is trained by updating weights using forward and backward propagation within a fixed number of epochs. After running Algorithm 1, it will be integrated to the *Intelligent Communication Assistant*. The purpose is to suggest the right need at each new state, as we will detail in the next subsection.

Intelligent Communication Assistant for LIS Patients

The designed ambient environment is able of delivering data using the devoted sensors. The latter will be the input of the *Intelligent Communication Assistant*, as described in Figure 5. For more simplicity, let us describe an example of the User Story (US). First, data is collected from the ambient environment. Second, the system analyzes the environment and suggests a need to the patient. Third, the patient receives a notification and depending on his mind, he can accept or reject the proposal. Forth, if he validates it, the communication system connects him directly to the caregiver.

For example, it is 08 AM, the patient gets up, and he may need some light to start the day. The *Intelligent Communication Assistant* analyzes the room state, calculates the reward, proposes the action with the highest reward and sends it as a notification to the patient. If the highly rewarded action is "Augmenting the light," the patient approves it as it corresponds to what he needs. The assistant connects him to the caregiver and saves

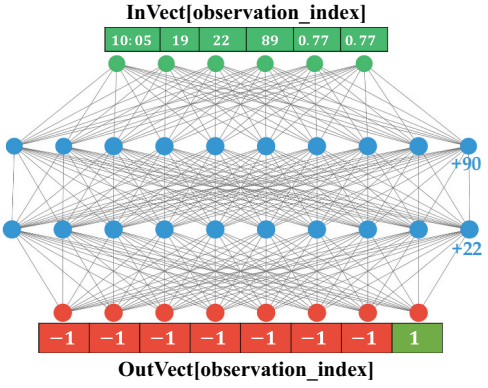


Figure 4. Example of the input and output vectors.

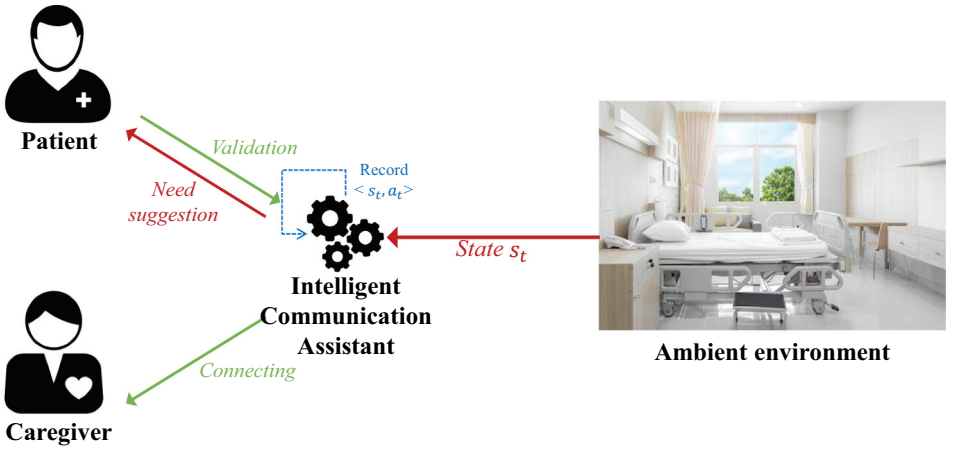


Figure 5. The architecture of the intelligent communication assistant's system.

the couple of (state, action), as detailed in Algorithm 2. At this phase, the caregiver is free to open the curtain or the light, depending on the patient's case and intentions.

Algorithm 2 Need suggestion
Input:Trained neural network
1: **procedure** SUGGESTION-TO-APPROVE(Trained Neural Network)▷ Function to integrate in the AAC system
2: **for each** New recorded state s_t **do**
3: Estimate the reward payoff of each action $\hat{r}_a$ by processing new s_t inputs using NN
4: Suggest the higher rewarded action to the Patient (notification)
5: **if** The patient accepts the suggested need **then**
6: Call the caregiver
7: Record the couple of (State, Action) i.e., (s_t,a_t)

Environment Description and Simulation

To feed Algorithm 1 with the required inputs, we need a time-recorded dataset that describes the environment of the patient at the moment of expressing a physiological need. The existing benchmarks do not cover the problem of physiological needs or patients room data. This explains why we generated synthetic data according to many logical assumptions that describe the behavior of the patient's overtime. In this section, we will present those assumptions and then describe the content of the dataset. This dataset was validated by experts.

Environment Representation

The first step consists in highlighting the environment states. Obviously, it is characterized by many features. The idea, here, consists in recording the state of the room at the time of need of expression. Let's suppose that the patient has expressed n needs after 1 day. He can choose one of the 8 actions that we described previously. Choices return only to him, he can ask for all of them, or he can just ask for sleep and spend the entire day sleeping at low luminosity. The number of actions per episode returns only to the patient's intents.

Imitating the reasoning of the patients by following some logical assumptions to generate data is a real must. Then, the generated data will describe the patient's behavior and habits. Table 2 explains some considered hypotheses. As a start, the expert team at Biware (hosting entreprise) tried to imitate the reasoning of the patient when he expresses one of the following needs: *Open the window, Close the window, Change the bed position, Move, Drink, Set the temperature, Sleep* and *Adjust the light*.

The chosen actions define the most basic physiological needs related to the homeostasis of the patient. They do not cover all the biological requirements for human survival, and we should search for more links between the environment and the patient's need. But for now, we will use them to validate our hypothesis.

Dataset Preparation

The dataset describes the patient behavior. We published it on Mendeley Data.[5] Each daytime is represented by an *episode*. Simply, Figure 6 gives an example of two episodes. The latter describe the same two episodes that are mentioned in Table 3.

Table 2. Some of the considered assumption in the data generation.

ID	Action: patient's need	Assumption
0	Open the window	The patient can open the window all day long between 7AM and 5PM.
1	Close the window	Considering that closing the window is at 5:00 PM in winter and 7:00PM in summer.
2	Change the bed position	The patient can change the position at any time of the day according to his intents.
3	Move	For example: the patient is used to move between 11:00AM to 12:00PM.
4	Drink	The patient may ask for a drink at any time and the bed must be in position 3 (The patient is awake).
5	Set the temperature	The average temperature in the patient's room should be between 18 and 20 degrees. If the patient wants some change in the aeration, he may ask this proposition.
6	Sleep	For instance: the patient may ask to sleep between 2:00 PM and 10:00 PM
7	Adjust the light	There are two possibilities: whether it's time to sleep so the patient will ask for standby lighting or the brightness level of the room is so low (less than 40) so the patient needs more light.

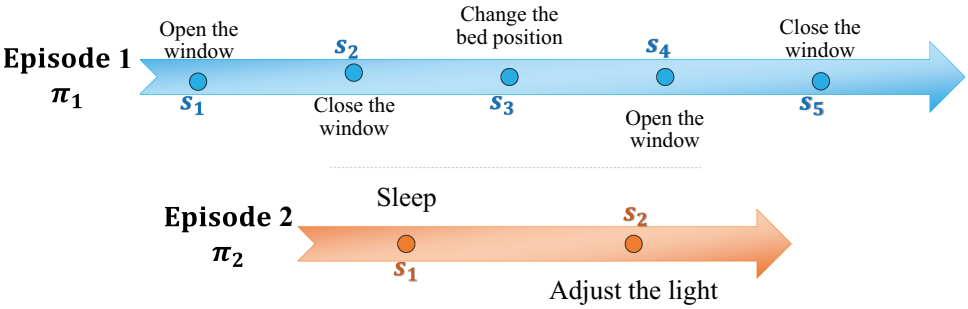


Figure 6. Example of two episodes.

In the first episode, there are five recorded states recorded on five different times $(t_1, ..t_5)$. The patient wants to open the window $(a_t = 0)$ two times on the same day, respectively, at 16:22 and at 18:50. Then, he decided to close it $(a_t = 1)$ at 17:12 and 17:56. Also, he Changed the bed position $(a_t = 2)$ at 17:36. In the second episode, there are only 2 recorded states, the patient asked for sleep $(a_t = 6)$ at 13:36 and he wants to adjust the light $(a_t = 7)$ at 18:56. As we can see, the dataset describes an unpredictable behavior that refers to the most important assumption that enunciates that needs refer to the patient's intent and mood. The ambient environment is changing over time, which explains the variation of the values of the environment states.

To this end, we succeeded in generating a dataset that imitates the behavior of the patient by following many logical assumptions to be as close as possible to the real need expression situations. Table 4 resumes the number of episodes (i.e., days). The number of observations is the total recorded needs during all the days. The purpose of associating 1800 i.e., 29% for the test dataset is to see the ability of generalization of the Intelligent Assistant after training it on 71% of the dataset.

Experimentation and Results

In this section, the results of the execution of the proposed algorithm are presented step by step. At first, we start by explaining the adopted Neural Network (NN) Model and its parameters. Then, we evaluate its performance.

Implementation

All the Deep Neural Network models act as a black box with a multi-layer nonlinear structure. Even if they showed their efficiency in many applications, they have, always, been criticized for their non-transparency and lack of

Table 3. Example of two episodes from simulated dataset.

Episode *Real Time*	Time Index *Exterior Temperature*	*Lighting percentage*	*Window*	State *Bed* state	Action a_t			*Room Temperature*
		16:22	19	22	89	0.77	0.22	0
		17:12	18	16	31	1.5	0.9	1
		17:36	20	15	72	1.74	0.83	2
		18:50	20	17	37	1.9	2.71	0
		18:56	18	17	48	1.69	0.37	1
		13:36	20	15	37	0.6	0.11	6
		18:56	18	15	24	0.43	0.32	7

Table 4. Dataset size.

Dataset	Total episodes: Days	Total observations
Train	2057	4500
Test	803	1800

prediction traceability. In this section, we will try to explain our reasoning in shaping a generative NN capable of giving a rational reward to the most fittable needs according to the environment.

The NN is composed of two layers of nodes and designed to learn to map examples of inputs to the desired outputs. Figure 4 summarizes the shape of the model. Table 5 resumes the chosen parameters. The input layer is composed of 6 nodes that describe the ambient environment's features. Two hidden layers are entailed respectively *100* and *32* nodes. The output layer has *8* nodes, each one of them involves estimating the reward of the related action.

On the other hand, Neural networks work depending on the chosen *activation functions*. For this purpose, the two hidden layers work corresponding to the values returned by the activation ReLu[6] function. The latter is chosen because of its sensitivity to the activation of the sum of the inputted values and avoid the vanishing gradient problem. Finally, we applied Algorithm 1 by putting the reward payoffs in the output layer for the purpose of regression.

Performance Evaluation

The designed Neural Network model aims to estimate the reward payoff of the new states. This section analyzes the behavior of the learning process by revealing the correctly suggested actions comparing to the real ones, and also by analyzing the performance of regression. To do so, we used the *test dataset* on the shaped NN after training it. We consider the highest reward estimation as a suggestion to the patient. Thus, We measured the *Matching* by considering the fact of associating the highest reward to the right action compared to the total number of actions i.e., number of observations (See Equation 11)

$$\text{Matching percentage} = \frac{100 * \text{Well suggested need}}{\text{Total number of needs}} \tag{11}$$

Table 5. Deep neural network parameters.

Layer	Number of nodes	Activation Function
Input	6	–
Layer 1	100	ReLu
Layer 2	32	ReLu
Output	8	–

Table 6. Percentage of giving the highest reward to the fitable action.

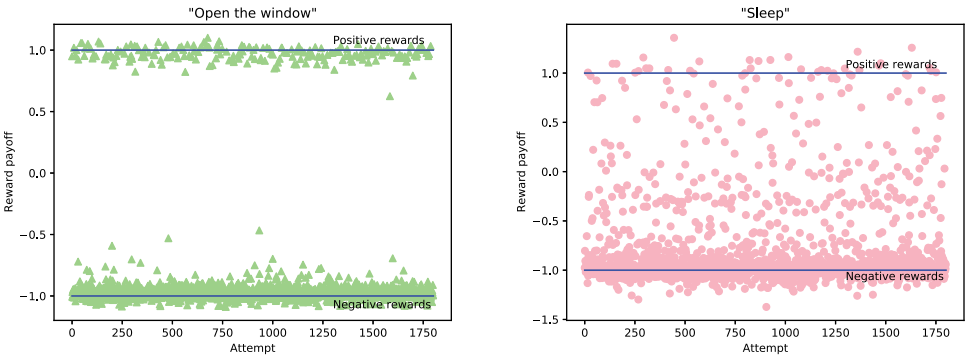
Action (need)	*Open the window*	*Close the window*	*Change the bed*	*Move*	*Drink*	*Set the temperature*	*Sleep*	*Adjust the light*
Total Number	200	200	200	200	200	400	200	200
Matching	100%	100%	98%	96.5%	88.5%	89.5%	56%	96%

This experimentation permits us to reveal some information about the well-predicted needs. Table 6 groups the result after processing the *train dataset* 200 epochs.

Obviously, the neural network succeeds in learning when the patient wants to *open or close the window*. Also, it predicts well the actions: *Change the bed position, Move* and *adjust the light*, as highlighted in Table 6. On the other hand, it decreases its precision when dealing with *Drink* and *Set the temperature* actions. Finally, it drops out its effectiveness when trying to understand the *Sleep* behavior of the patient. We conclude by saying that very well predicted needs are directly related to the environment features that's why the neural network succeeds in finding the relationship between them thanks to weights and biases calculation. On the other hand, predicting that the patient wants to sleep isn't an easy task because it is related to only two features (position of the bed and lighting percentage). In the simulation assumption, the patient relies on the caregiver to be able to ask for a Move, and the caregiver is not always available to answer to his requests. That's why the model arrives to detect the relationship between the time of moving and the need.To this end, we can conclude that needs which are dependent in the environment are the luckiest to be well predicted and suggested to the patient, contrast to those related to his intent and mood. To conclude, our intelligent assistant is capable of suggesting the right need with a 90.56% of Matching (the average of all the matching percentages that figured in Table 6).

At this phase, we want to study the reward regression behavior over time. Reward regression consists in predicting the payoff of the actions by learning from the environment features. To evaluate the results, the comparison should be between the real values and the estimated values. Supposing that true reward function R_E is available for purposes of evaluation, the accuracy is the closeness of the learned reward function $\hat{R}_e$ compared to the real reward payoff R_E like follows $\left\|R_E - \hat{R}_e\right\|$. To study the performance of the estimated payoffs, plotting the regressed rewards gave us a better visibility. We compared the reward regression of the following two needs: *Open the window* that has 100% Matching and *Sleep* that has only 56% of right suggested needs, and we grouped them in Figure 7.

As illustrated, the first sub-plot is perfectly shaped. The model arrives to associate the positive and negative reward for all the tested values. Contrast to the second subplot, in which values are scattered between −1.5 and +1.5. Lines



(a) Open the window (b) Sleep

Figure 7. Comparing needs reward regressions.

indicates the real needs' reward payoff that we used in the training process since we associated +1 to the right answer and −1 to the wrong one. The distance between the points and that line illustrates the error of the prediction.

To this end, we showed the efficiency of the proposed algorithm and model in suggesting the right action at the right time. Even if regressed values are not conformable to the real reward since the input features are very close. The higher reward payoff is, in most of the time, attributed to the real action. The idea of the proposed Algorithm showed its efficiency when testing with the synthetic database.

Conclusion

In this paper, we propose a new *Intelligent Communication Assistant* that succeeds in detecting eight needs of the LIS patient, which are: *Open the window, Close the window, Change the bed position, Move, Drink, Set the temperature, Sleep* and *Adjust the light*. We discovered that some needs are harder to predict like sleeping, and others are predicted with very high precision like opening the window. The system is able of suggesting the right need with a 90.56% of Matching (the average of all the Matching percentages).

This article presents the training DIRL algorithm using MaxEnt which consists in learning from the history of the behavior i.e., the time-recored ambient environment' states. The purpose is to regress the reward payoff of the new states and suggest the highest recorded action to the patient via a new communication pipeline. To do so, we proposed a new architecture for the *Intelligent Communication Assistant* that sends notifications to the caregiver if the patient needs something. The latter can be integrated in one of the existing AAC system.

This solution can be useful for other things like monitoring health status and sending notifications to the caregiver. Besides, it can treat more needs depending on the ambient environment's fed data. As

a perspective to this research work, we aim to integrate this module on a real AAC system and try to see its impact on different categories of patients. Our starting point is the work proposed by (Scobee, 2019) who have taken a step toward addressing the incorporation of safety into the context of IRL. The goal is to ensure that our system is safe to use and deploy, which is a critical element to the acceptance and adoption. The challenge here is to propose a new system able of dealing with different types of patients, and that is able of discovering the behavioral change without decreasing the matching percentage (Accuracy). This means that sensitivity analysis is very important because the considered hypothesis cannot represent all the patients. That is why we aim to study the impact of the unpredictable behavior on the Matching of reward regression. Thus, more needs, more precision, and exactitude are required because the solution will be used to understand and model the needs of a very sensitive category of patients.

Finally, the proposed model should be linked to a real ambient room by increasing context-awareness. Thus, the full architecture should be implemented and tested. Also, studying the security and privacy of the ambient environment will improve the reliability of the solution to be ready for use. Finally, the system will be more helpful when it treats more needs. Thus, treating more needs belonging to the physiological level proposed by (Maslow 1958, 1981) will add new actions to the system.

Notes

1. https://theeyetribe.com/
2. https://eyegaze.com/
3. https://www.eyecontrol.co.il/
4. https://www.gazept.com/
5. https://data.mendeley.com/datasets/z3fvh52px9/1
6. ReLu is an activation function that acts as a linear function for values that are greater than 0 and associates zero to negative values. It's a nonlinear function that allows learning complex relationships between data by respecting the function: $g(z) = max\{0, z\}$ with z the weighted biased sum of the inputted nodes.
7. https://biware-consulting.com/

Acknowledgments

This project was done in collaboration with *Biware Consulting company*[7]. The dataset was proposed by Miss Rahma Hellali at her master thesis internship. She was mentored by their consulting staff who were very engaged toward this topic since they approved the dataset that we used in our work and validate it with experts. They, also, validate the different obtained results. We really acknowledge their efforts in providing us with the necessary information.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Khaoula Hantous http://orcid.org/0000-0001-7190-5657
Lilia Rejeb http://orcid.org/0000-0002-5740-1556
Rahma Hellali http://orcid.org/0000-0002-1306-4682

References

Abbeel, P., A. Coates, M. Quigley, and A. Y. Ng. 2007. An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems* 19: 1–8. https://proceedings.neurips.cc/paper/2006/file/98c39996bf1543e974747a2549b3107c-Paper.pdf.

Abbeel, P, A. Coates, A. Y. Ng *et al* . 2010. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29 (13):1608–1639. doi:10.1177/0278364910371999.

Arora, S., and P. Doshi. 2018. "A survey of inverse reinforcement learning: Challenges, methods and progress ." *arXiv preprint arXiv:1806.06877*. https://arxiv.org/abs/1806.06877

Arora, S., and P. Doshi. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* 297:103500. doi:10.1016/j.artint.2021.103500.

Bauer, G., F. Gerstenbrand, and E. Rumpl. 1979. Varieties of the locked-in syndrome. *Journal of Neurology* 221 (2):77–91. doi:10.1007/bf00313105.

Bellman, R. 1957. A Markovian decision process. *Indiana University Mathematics Journal* 6:679–684. doi:10.1512/iumj.1957.6.56038.

Chareonsuk, W., S. Kanhaun, K. Khawkam, and D. Wongsawang. 2016. "Face and eyes mouse for ALS patients." In *2016 Fifth ICT international student project conference (ICT-ISPC)* May 27-28, 2016 Nakhonpathom, Thailand, IEEE, 77–80 doi:10.1109/ICT-ISPC.2016.7519240. .

Chen, X.-L., L. Cao, Z.-X. Xu, J. Lai, and C.-X. Li. 2019. A study of continuous maximum entropy deep inverse reinforcement learning. *Mathematical Problems in Engineering* 2019 1–8 doi:10.1155/2019/4834516 .

Coronato, A., M. Naeem, G. De Pietro, and G. Paragliola. 2020. Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine* 109:101964. doi:10.1016/j.artmed.2020.101964.

Ivanov, S., and A. D'yakonov. 2019. "Modern deep reinforcement learning algorithms ." *arXiv preprint arXiv:1906.10025*.

Kartakis, S., V. Sakkalis, P. Tourlakis, G. Zacharioudakis, and C. Stephanidis. 2012. Enhancing health care delivery through ambient intelligence applications. *Sensors* 12 (9):11435–50. doi:10.3390/s120911435.

Light, J., D. Mcnaughton, D. R. Beukelman, S. Fager, M. Fried-Oken, T. Jakobs, and E. Jakobs. 2019. Challenges and opportunities in augmentative and alternative communication: Research and technology development to enhance communication and participation for individuals with complex communication needs. *Augmentative and Alternative Communication* 35 (1):1–12. doi:10.1080/07434618.2018.1556732.

Loja, L. F. B., R. de Sousa Gomide, F. Freitas Mendes, R. Antonio Gonçalves Teixeira, R. Pinto Lemos, and E. Lúcia Flôres. 2015. "A concept-environment for computer-based augmentative and alternative communication founded on a systematic review." Sep. https://scielo.figshare.com/articles/A_concept-environment_for_computer-based_augmentative_and_alternative_communication_founded_on_a_systematic_review/7518137/1 .

Markov, A. A. 1954. *Theory of Algorithms*. TT 60-51085. Academy of Sciences of the USSR. https://books.google.tn/books?id=mKm1swEACAAJ.

Maslow, A. H. 1943. A theory of human motivation. *Psychological Review* 50 (4):370–96. doi:10.1037/h0054346.

Maslow, A. H. 1958 A Dynamic Theory of Human Motivation. Understanding human motivation. (Howard Allen Publishers)26–47 doi:10.1037/11305-004

Maslow, A. H. 1981. *Motivation and personality* Prabhat Prakashan. Addison-Wesley Educational Publishers Inc.

Newell, A., S. Langer, and M. Hickey. 1998. The rôle of natural language processing in alternative and augmentative communication. *Natural Language Engineering* 4 (1):1–16. doi:10.1017/S135132499800182X.

Ng, A. Y., S. Russell, et al. 2000. Algorithms for Inverse Reinforcement Learning Proceedings of the Seventeenth International Conference on Machine Learning June 29 - July 2, 2000 San Francisco, CA, USA. (Morgan Kaufmann Publishers Inc.):663–670.

Patents, J. 2020. "Patents assigned to TOBII AB." https://patents.justia.com/assignee/tobii-ab. (Last checked on May 11, 2021).

Pieter, A., and A. Y. Ng. 2010 Inverse Reinforcement Learning . *Encyclopedia of Machine Learning*, 554–558 978-0-387-30164-8 . Boston, MA: Springer US. doi:10.1007/978-0-387-30164-8_417.

Russell, S. 1998. "Learning agents for uncertain environments." In *Proceedings of the eleventh annual conference on Computational learning theory* July 24 - 26, 1998 New York, NY, United States, 101–03 https://people.eecs.berkeley.edu/~russell/papers/colt98-uncertainty.pdf.

Scobee, D. R. R. 2019 Approaches to Safety in Inverse Reinforcement Learning. Berkeley: University of California. https://escholarship.org/uc/item/6j34r5tp

Smith, E., and M. Delargy. 2005. Locked-in syndrome. *BMJ* 330 (7488):406–09. doi:10.1136/bmj.330.7488.406.

Sutton, R. S., A. G. Barto, et al. 1998. *Introduction to reinforcement learning* 1st . vol. 135. Cambridge, MA, USA: MIT press Cambridge. https://dl.acm.org/doi/book/10.5555/551283

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement learning: An introduction*. Cambridge, MA, USA: A Bradford Book.

Tatler, B. W., D. Witzner Hansen, and J. B. Pelz. 2019. Eye Movement Recordings in Natural Settings. *Eye Movement Research*, 549–592. Springer International Publishing. 10.1007/978-3-030-20085-5_13.

Tay, L., and E. Diener. 2011. Needs and subjective well-being around the world. *Journal of Personality and Social Psychology* 101 (2):354–65. doi:10.1037/a0023779.

Wahba, M. A., and L. G. Bridwell. 1976. Maslow reconsidered: A review of research on the need hierarchy theory. *Organizational Behavior and Human Performance* 15 (2):212–40. doi:10.1016/0030-5073(76)90038-6.

You, C., L. Jianbo, D. Filev, and P. Tsiotras. 2019. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems* 114:1–18. doi:10.1016/j.robot.2019.01.003.

Yuxi, L. 2017. Deep reinforcement learning: An overview. *CoRR*. http://arxiv.org/abs/1701.07274.

Yuxi, L. 2018. Deep Reinforcement Learning. *CoRR*. http://arxiv.org/abs/1810.06339.

Zhifei, S., and E. Meng Joo. 2012. "A review of inverse reinforcement learning theory and recent advances." In *2012 IEEE Congress on Evolutionary Computation*, June 10-15, 2012 (IEE) Brisbane, QLD, Australia, 1–8 doi:10.1109/CEC.2012.6256507.

Ziebart, B. D., A. L. Maas, A. K. Dey, and J. Andrew Bagnell. 2008b. "Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior." In *Proceedings of the 10th international conference on Ubiquitous computing*, September 21-24, 2008 Seoul Korea, 322–31 doi:10.1145/1409635.1409678.

Ziebart, B. D., A. L. Maas, J. Andrew Bagnell, and A. K. Dey. 2008a Maximum entropy inverse reinforcement learning AAAI Conference on Artificial Intelligence July 13–17, 2008. Chicago, Illinois. 8:1433–1438. Chicago, IL, USA: AAAI Press. https://dl.acm.org/doi/10.5555/1620270.1620297