Intelligent Environments 2018
I. Chatzigiannakis et al. (Eds.)
© 2018 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-874-7-38

Reinforcement Learning Based Approach for Smart Homes

Jose Mirra [a], Fábio Silva [a], Cesar Analide [a]
[a] *ALGORITMI Centre, University of Minho, Portugal*

Abstract. Smart Homes are environments that automate action and adapt themselves to user behaviours. In this sense, it is necessary to employ learning strategies to allow Smart Homes to truly become intelligent, in a sense that they anticipate needs and actions. This requires constant monitoring of environments, users and their actions, as well as, non-supervised dynamic learning strategies.

The purpose of this work is to develop a system capable of taking the best action possible based on its environment. In this document, we present a reinforcement learning approach to automate lights and appliances in a Smart Home environment. An intelligent agent perceives the ambient and the past interactions of the user with the home in order to learn what is the best action to perform, which action has a certain reward associated in order to inform the agent his behavior. A reinforcement learning algorithm learns a policy for picking actions by adjusting its weights through gradient descent using feedback from the environment.

Keywords. Reinforcement Learning, Smart Home, Machine Learning, Intelligent Agent

1. Introduction

Smart Homes are being more coveted nowadays, a recent research of [1] has shown a gain interest on Smart Homes but a decrease in home automation over time, this is simply because Smart Home technologies can do more than automate a house. This lead to more and more companies diving in this area of business, for instance, tech giants like Google LLC, Samsung, Apple and more [2]. The need of extending capabilities is essential to make a difference in the market.

Smart Home is defined as a living or working space that interacts in a natural way and adapts to the occupant. Adaptation refers to the fact that it learns to recognize and change itself depending on the identity and activity undertaken by the occupant with minimal intervention [3]. Hence, a Smart Home agent must be able to predict the mobility patterns and device usages of the inhabitants. The Smart Home behaves as a rational agent, perceiving the state of the home through sensors and acting on the environment through actuators. The goal of the Smart Homes is to maximize comfort and safety, optimize energy usage and eliminate strenuous repetitive activities.

A single day in the life of a person consists in a numerous set of actions. These actions, over a period of time, can be learned by an intelligent system and prove useful to predict future actions. For instance, the blind case. A person waking up at 8 am opens the shutter in the bedroom, an intelligent system could learn this pattern and switch shutter

on the room at a particular time predicting the waking time of the user. Another possible scenario would be the opposite, the system predicts when the shutters are closed. A system could learn to do this by analyzing the users patterns.

In this article, a deep learning system is proposed to learn user habits within a Smart Home and predict the best interval to activate actions. The system uses a reinforcement learning technique to predict the reward for each 15-minute interval in a day for a given action. The input to this system is a simulated historical data, which contains user past interactions with the system. The system makes use of a deep learning technique and a policy technique to give rewards to the learning system as it tries to learn user habits with each action that takes place along the day.

The multi-agent system and algorithms proposed and methodology along with the results obtained are discussed in the next sections of this article document.

2. Related Work

Existing projects towards Smart Homes use mostly a multi-agent system (MAS) methodology. In multi-agent systems, agents can additionally communicate and coordinate with each other as well as with their environment. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [4] . MAS provides a valuable framework for intelligent control systems to learn building and occupancy trends. [5] . However existing systems use mostly a pattern mining technique [6] to determine what can be the next action. An example of this is the MavHome project by Cook et al. [7], an home that acts as a rational agent which aims to maximize comfort and productivity of its inhabitants and minimize operation cost. In order to achieve this goals, the house must be able to predict, reason about, and adapt to its inhabitants.

Therefore, some systems use a reinforcement learning approach, Che et al. (2010) use an event processing tool to handle the events from the home automation devices, prediction algorithms to predict the next action and reinforcement learning to decide which actions are suitable to be automated [8].

In recent past, with the advance of deep learning, a new level of reinforcement learning has surged Deep reinforcement learning. Although there are challenges, impressive applications has appeared. Mnih et al. (2013) from the University of Toronto show how a computer can beat a human in an old Atari video game using deep reinforcement learning in Tensorflow. Tensorflow is an open-source machine learning framework. It used a Q-Learning model combined with a Convolutional Neural Network to predict the next action, given the image of a game at that instant [9].

More recently, Gokul et al. (2016) present the first deep reinforcement learning model for home automation system. The system makes use only of images to learn the users needs using Deep Q-Learning to make the system understand and learn the user patterns and actions without the need for intervention of a user [10]. The results are similar to ours, it also delineates rewards for each action for different time intervals in a day based on a robust reward function based on Q-Function. However, the lack of user intervention leads to system without user feedback, in this way, the feeling of control is not maintain. According to Balta-Ozkan et al. (2013) there are some impasses regarding the social embrace of this technology, being loss of control and apathy one of the main concerns [11]. Our approach aims to avoid these social barriers.

In this paper, Tensorflow deep reinforcement learning is used in the context of Smart Homes, using a gradient policy to make the system understand and learn the user patterns and actions. The data collected by the system is controlled by the user, providing a feeling of control. In addition, the user has the power to decide which action he wants to be automatized of all available in the system.

3. Reinforcement Learning

Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. The primary aim is to cast learning as a problem involving agents that interact with an environment, sense their state and the state of the environment, and choose actions based on these interactions. In reinforcement learning the agent comes pre-equipped with goals that it seeks to satisfy. These goals are embodied in the influence of a numerical reward signal on the way that the agent chooses actions, categorizes its sensations and changes its internal model of the environment. Despite the obvious connection of these terms to behavioral psychology, some of the more impressive applications of reinforcement learning have been in computer science and engineering applications [12].

Reinforcement learning lies between supervised and unsupervised learning. It makes use of sparse and time-delayed labels - rewards. The system at each stage tries to maximize the reward.

This approach will not consider states, this means that rewards for an action is not conditional on the state of the environment. All we need to focus on is learning which rewards we get for each of the possible actions, and ensuring we choose the optimal ones, this is called learning a policy and we will discuss our policy in section 3.1.

3.1. Policy Gradient

The simplest way to think of a Policy gradient network is one which produces explicit outputs. In the case of our case study, we dont need to condition these outputs on any state. As such, the network will consist of just a set of weights, with each corresponding to each of the possible time of the day to do an action, being turn the lights on, shutter, and so on, and will represent how good our agent thinks it is to do the respective action in that interval. Weights are all initialized at the same value, being all intervals equal to the agent at the initial state.

To update our network, we will simply use an e-greedy policy. This represents that our agent will choose the action that corresponds to the largest expected value, but occasionally, with e probability, it will choose one interval randomly. This randomness is crucial, it allows the agent to try out each of the different intervals of the day to continue to learn more about them, without it, the agent would ignore potential better intervals than the one chosen. Once our agent has taken an action, it then receives a reward of either 1 or -1. With this reward, we can then make an update to our network using the policy loss equation 1.

$$loss = -log(\pi) * A \tag{1}$$

In equation 1, A is advantage, and is an essential aspect of all reinforcement learning algorithms. It corresponds to how much better the agent action was than the baseline, being 0. Intuitively, a positive reward indicates a increase weight to that specific interval, and a negative reward, a decrease weight.

The parameter π is our policy. In this case, it corresponds to the chosen interval weight.

This loss function allows us to converge to the best interval to do the respective action. The agent will be more or less likely to pick that action in the future. By taking actions, getting rewards, and updating our network in this circular manner, we will quickly converge to an agent that can solve our problem.

In order to predict the best interval of the day to execute possible actions. We developed multiple agents, each with their respective weights and rewards. These agents run simultaneous, being possible by our chosen machine learning framework, Tensorflow.

3.2. Learning Agents

Agents are responsible for perceiving the past interactions of the user and, based on that information, choose the best time of the day to perform the action. Each agent, one for each possible action, take his decision according to what he thinks it has the most reward.

To accomplish that, we establish the agent updating procedure. With the policy defined, we use the Tenserflow library. It provides a set of classes and functions that help train models. In this work, we use the *GradientDescentOptimize* to train the reinforcement learning algorithm.

The parameter *Chosen interval* represents the interval chosen by the agent. Training with a gradient descent method. The agent is updated by moving the value for the selected action toward the received reward.

The agent will be trained by the actions taken in the environment, and receiving positive or negative rewards. Using rewards and actions, it allows us to know how to properly update our network in order to more often choose actions that will yield the highest rewards over time.

3.3. Reward Function

Reward functions describe how the agent ought to behave. In other words, they have normative content, stipulating what we want the agent to accomplish. In our work, we have multiple distinct actions that we want to accomplish, therefore, we created different agents. Different reward functions were needed, one for each possible action in our environment.

When the agent does the right action in the right interval, we reward it with a positive reward, otherwise, in other all actions possible, we reward it with a negative reward. There are only two possible rewards, this is called a binary reward function, where we specify a single action to be rewarded in a specific time of the day:

$$R(a1, interval) = 1$$

$$R(a2..n, interval) = -1$$

For example, when the agent predicts the best hour to turn on the lights, if it chooses to light them when the user usually turn them on ($a1$) it receives a positive reward, otherwise ($a2..n$), receives a -1.

4. Experiments

In this section, we present our approach. The architecture proposed, along with algorithm are discussed in this section.

4.1. Architecture

The figure shown in figure 1, represents our system architecture. A user interacts with an environment daily, these daily interactions are perceived and stored in a data set.

In order to answer the problem defined, which focuses in the importance of the user in an automation environment, it is needed to establish a solid connection between the automation system and the user. With this in mind, we developed a multi-agent system, with distinct agents. This multi-agent system is defined as Logical Unit in this paper.

The Logical Unit is responsible for the decision making in our system. It contains two types of agents, intelligent agents, and the interface agents. Interface agents are responsible for communicating with the user and the system. The system is capable of suggesting the user of the action chosen by our intelligent agents, and the user can express his opinion about the decision of the agents. This results in our system converging to the user personal preference actions. Interface agents are vital to accomplishing this, as they serve as intermediates between the system and the user.

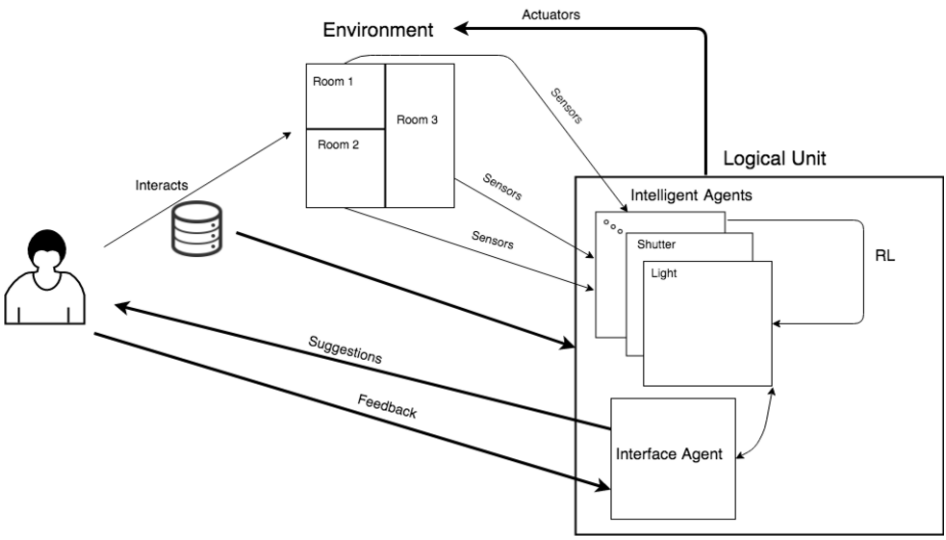


Figure 1. System Architecture

Intelligent agents can view and perceive the environment through sensors and act upon that environment through effectors. The environment is composed by different

rooms, these rooms have multiple sensors regarding the appliance. The logical unit has multiple intelligent agents, one for each appliance in the environment, lights, shutters, temperature and others. These agents perceive their respective sensor in the different rooms and after the deep reinforcement learning process, they actuate through actuators in the ambient. For instance, the shutter agent, is responsible for the automation of the shutters in the different rooms.

Reward functions are dependent of the user pattern, present in historical data sets, in account which action the user took in that instant, if it correspond to the agent decision, a positive reward is granted. The reward function associated with each action can drastically change after receiving the user feedback, indicating the possibility of adapting to the user preference.

With the architecture presented, it is possible to answer the problems raised in this paper. It is capable of converging to different usage patterns and discern between different contexts in the form of user preferences. The possibility of communication between the user and the system is the principal distinct factor between this work and the previous one studied in the related work.

4.2. Algorithm

The reinforcement learning algorithm present in the logical unit is responsible for making the intelligent agents intelligent, it allows them to learn about the environment and acting on them, these actions are rewarded in order to converge to an optimal automation. Our ultimate goal is to know when to do a respective action, for example, in which time of the day we should open or close the shutters. First, we need to define the global variables and initialize them, define the number of intervals in the day, initialize the weight and define the loss function, initialize total number of episodes to train agents on, initialize total reward for intervals for each agent and define the chance of taking a random interval e. Then we proceed to define the multiple reward functions, one for each intelligent agent procedure. Lastly, agents will train by taking actions in our environment through the following procedure in algorithm 1.

Algorithm 1 Deep Reinforcement Learning

```
1:  procedure TENSERFLOW SESSION TO OPEN SHUTTERS
2:      tf.Sessions() as sess:
3:      dataSet ← Historical Data
4:      Initialize loop counter i=0
5:      while Number of Days do
6:          Initialize loop counter j = 0
7:          while Number of Episodes do
8:              if random number < e then
9:                  interval ← Random interval
10:             else
11:                 interval ← chosen interval
12:             reward Open ← Reward Function for interval
13:             Update the network and the Reward
14: procedure TENSERFLOW SESSION TO X ACTION
15:     Same procedure as open shutters Session, Reward Function differs.
```

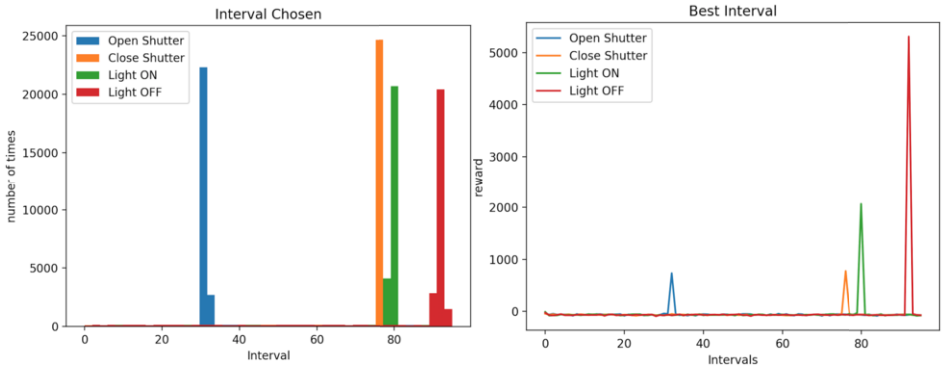
These procedures are replicated to the different intelligent agents present in the system, with an exception for reward functions. Each intelligent agent has is own reward function as is own reward variable. This is absolutely needed since each action is independent of other actions present in the system, the user has the power to decide each action wants to be automatized, thus the need of independence on these intelligent agents.

5. Results

In this section, the results obtained are discussed for the case study presented. For experimental purposes, we simulate a user interaction with a system, based on real events of daily activities. It is considered the light and shutter appliances. The environment is a home environment with 2 different rooms.

We have created two distinct users, each one residing in a different room. In order to replicate a real situation, the data were simulated based in a Gaussian deviation, in other words, the data-set contains some noise, adding some challenge to the algorithm to find a pattern. In the results presented in this section, we consider 96 intervals, each one of 15 minutes, representing a day.

Agents are trained using 30 days of data. The training procedure consists of 1000 repetitions of daily data. In order to explore all intervals defined, the probability to choose a random interval was delineated to 20 percent. The following figures, figure 2 and 3 represents the decision making on both rooms, respectively in 1 and 2.



(a) Agent chosen action

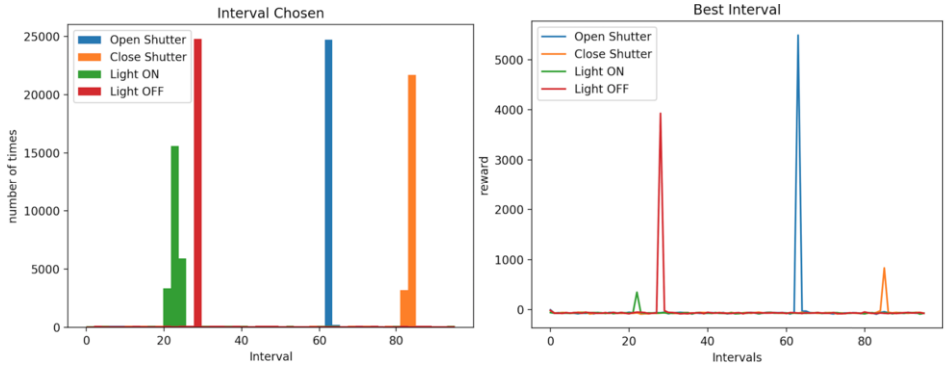
(b) Rewards for the different intervals and actions

Figure 2. Room 1

Room 1 is inhabited by a day laborer. It can be clearly seen in figure 2a, that the agent inclines significant more to a determinate interval than others, when considering the action. This is caused by the neural network prediction. For example, the agent predicts that the interval 32 is best to open the blinds in room 1, representing 8:00 am, which makes sense for the day laborer who wakes up around this hour.

For each action chosen, the agent will confiscate his reward. On figure 2b, we can see the reward associated to each action at a certain interval. In this case, our algorithm prediction is according to the system reward. Seeing the shutter example once more,

interval 32 is the interval with most reward for opening the shutters, solidifying our agent prediction.



(a) Agent chosen action (b) Rewards for the different intervals and actions

Figure 3. Room 2

Room 2 is inhabited by a nocturne laborer. Figure 3a represents the interval predicted by the agent. It differs significantly when compared to room 1, with this, it can be affirmed that the system is capable of adaptation to different behaviors in the environment.

Figure 3b represents once more, the reward associated with each action at a certain interval. Some actions have a bigger rewards than others, for example, our system is more confident at the hour of turning the lights off comparing to turning them on. This is simply because of the variation of the light sensor in the user past activities. The more unpredictable the user is, the less confidence the system has.

In order to show the impact of the user on the system, lets remind the day laborer example in room 1. Figure 4 represents the updated rewards after the user disagrees with the system decision on turning the light off in 2b. As expected, the user has a great impact in the system decision, giving more weight to the user feedback than to the historical data in the reward function.

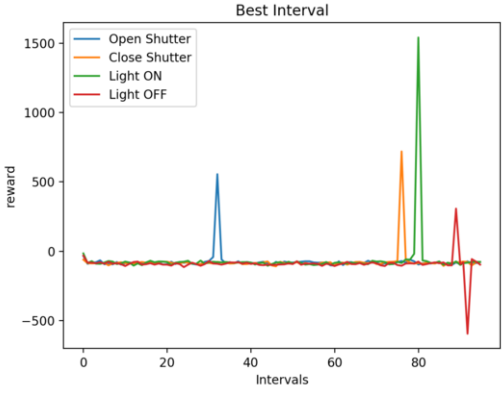


Figure 4. Rewards for the different intervals and actions after feedback in Room1

In this section, we simulate a user interaction with a system based on a Gaussian distribution, to provide some variance in the intervals chosen by a user for a determined action, however, using real data, we expect more unpredictability. This result in more variation in our reward by interval graph, as the highest reward value is not so inclined to one specific interval, difficulting the identification of time intervals where these actions are more appropriate. To avoid such variance, pre-processing data is expected to be needed, alongside with a more robust and complex reward function, tweaked to minimize the confusion.

For now on, we aim to develop a capable system to be used in a real world environment.

6. Conclusion

In this paper, a reinforcement learning model approach for Smart Home is introduced. Unlike the previous reinforcement learning approaches, this paper highlights the role of the inhabitant to combat the social impasses of Smart Homes. With the help of deep learning and neural networks, the ability of the system to predict the subsequent action given the state of the environment and the user pattern is successfully demonstrated based on non-supervised techniques.

The system was tested based on its ability to predict the best 15 minute interval present in day for lights and blinds in different rooms present in a home. User Feedback was also been tested and it demonstrates how influential it is upon the system. More research work has to be performed to generalize this system to all appliances. It was demonstrated that this model for home appliances, but it can be extended to others environments, either public or private.

It is intended to increase the complexity of the reward functions, being more dependent of the user feedback and environment sensing instead of being totally dependent of their user pattern. In this way, the aim is to help the user to avoid wrong habits, directing them to a routine they desire. Also, external data from public Application programming interfaces (API's) are planned, allowing this system to not be restricted to sensors to perceive the environment.

Acknowledgement

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundao para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within the Project Scope UID/CEC/00319/2013.

References

[1] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. A re-view of Smart Homespast, present, and future.IEEE Transactions on Systems, Man, andCybernetics, Part C (Applications and Reviews),42(6):11901203,2012.

[2] Rita Yi Man Li, Herru Ching Yu Li, Cho Kei Mak, and Tony Beiqi Tang. Sustainable smarthome and home automation: Big data analytics approach.2016

[3] Aditi Dixit and Anjali Naik, Use of prediction algorithms in smart homes. International Journal of Machine Learning and Computing,4(2):157,2014

[4] Stuart Russell and Peter Norvig.Artificial Intelligence: A Modern Approach. Prentice Hall,second edition,2003. ISBN0137903952

[5] Laura Klein, Jun-young Kwak, Geoffrey Kavulya, Farrokh Jazizadeh, Burcin Becerik-Gerber,Pradeep Varakantham, and Milind Tambe. Coordinating occupant behavior for buildingenergy and comfort management using multi-agent systems.Automation in construction,22:525536,2012.

[6] Diane J Cook, Michael Youngblood, Edwin O Heierman, Karthik Gopalratnam, Sira Rao,Andrey Litvin, and Farhan Khawaja. Mavhome: An agent-based smart home. InPer-vasive Computing and Communications,2003.(PerCom2003). Proceedings of the First IEEEInternational Conference on, pages521524. IEEE,2003.

[7] Sajal K Das and Diane J Cook. Designing smart environments: A paradigm based on learning and prediction. In International Conference on Pattern Recognition and Machine Intelligence, pages 8090. Springer, 2005.

[8] Natalie Kcomt Ch e, Niels Pardons, Yves Vanrompay, Davy Preuveneers, and YolandeBerbers. An intelligent domotics system to automate user actions. In Juan Carlos Au-gusto, Juan M. Corchado, Paulo Novais, and Cesar Analide, editors,Ambient Intelligenceand Future Trends-International Symposium on Ambient Intelligence (ISAmI2010), pages201204, Berlin, Heidelberg,2010. Springer Berlin Heidelberg. ISBN978-3-642-13268-1.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[10] Vignesh Gokul, Parinitha Kannan, Sharath Kumar, and Shomona Gracia Jacob. Deep q-learning for home automation.International Journal of Computer Applications,152(6):15,2016.

[11] Nazmiye Balta-Ozkan, Rosemary Davidson, Martha Bicket, and Lorraine Whitmarsh. Socialbarriers to the adoption of Smart Homes.Energy Policy,63:363374,2013.

[12] Richard S Sutton and Andrew G Barto.Reinforcement learning: An introduction, volume1.MIT press Cambridge,1998.