A Cooking Assistance System for Patients with Alzheimers Disease Using Reinforcement Learning

Haipeng Chen[1] and Yeng Chai Soh[2]

[1]Joint NTU-UBC Research Center of Excellence in Active Living for the Elderly (LILY)
[2]School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore

{chen0939, ecsoh}@ntu.edu.sg

Abstract

With the rapid increase of population with Alzheimer's disease, it has been more and more costly for the society to provide personal care to the patients. Artificial intelligence (AI) technologies, which are believed to significantly reduce the personal care cost, have been widely introduced to assist home care to the elderly people. In this paper, we design a reminder agent to help the patients with Alzheimer's disease in cooking tasks. The reminder agent divides the cooking task into a set of subtasks, and optimizes the time a reminder is released to the patient in each subtask. The cooking procedure is formulated as an MDP with continuous action space. Due to the large scale of the formulated MDP, we propose a solution to the MDP based on reinforcement learning.

Keyword: Reminder agent, MDP, reinforcement learning

I. Introduction

This In 2016, it is estimated that nearly 44 million people have Alzheimer's or a related dementia worldwide, and the global cost of caring Alzheimer's and dementia is estimated to be 605 billion, which is equivalent to 1% of the entire world's gross domestic product [1]. This situation is growing worse, as shown in Figure 1, the number of people affected by Alzheimer's is expected to grow to around 100 million in the year 2050.

Due to the cost efficiency and high quality care, AI technologies have been gradually adopted to assist daily home care to the patients with Alzheimer's. Cooking has been a critical, and yet challenging task for the patients with Alzheimer's (in the following, we will simply refer to them as ``patients"). Take Spaghetti as an example (Figure 2), there are several steps to successfully cook Spaghetti, which is complex to the patients without any assistance. In this paper, we propose a cooking assistance agent, which gives guidance to the patients while cooking. More specifically, our agent breaks down a cooking task (e.g., cooking Spaghetti) into several subtasks, and provide a reminder of what to do to the patient at a proper time during the subtask.

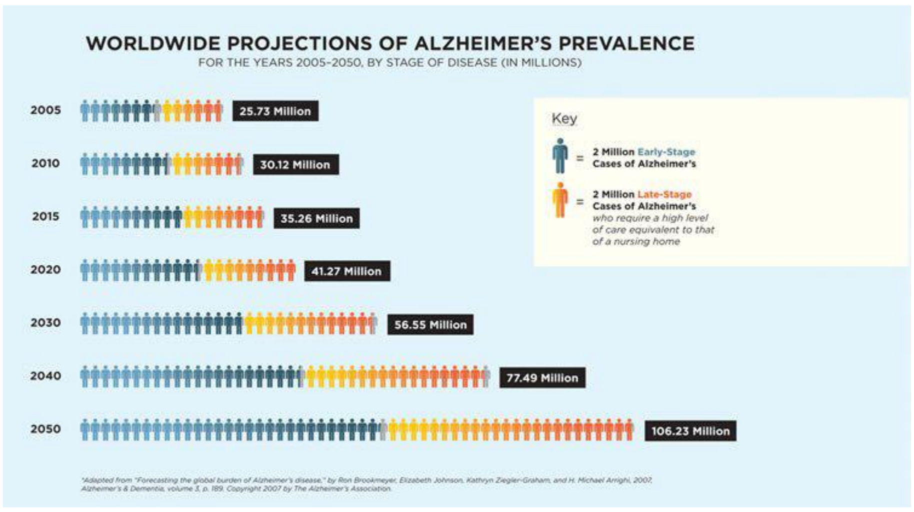

Figure 1 Statistics of Alzheimer's Disease

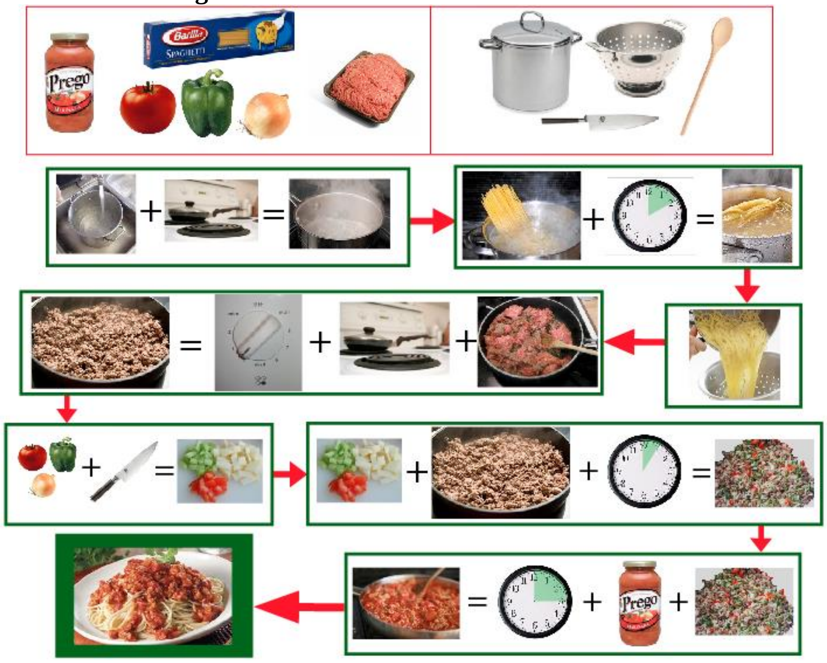

Figure 2 Steps of Cooking Spaghetti

We make two key contributions in this paper. First, we construct a formal model to formulate the cooking assistance task as an MDP. The state of the MDP is the time (or estimated time) the patient spends on each task, which reveals the mentality status of the patient. The action is to decide, at the beginning of each subtask, when to release a reminder to the patient. To our knowledge, there are no previous papers formulating the cooking assistance procedure as an MDP.

Our second key contribution is a solution to the formulated MDP based on inverse reinforcement learning. We show that the formulated MDP has continuous (and thus infinitely large) state and action spaces, which makes it very challenging to solve the MDP. In this paper, we propose a solution which 1) employs function approximation approach to represent the value function, and 2) a policy gradient method which directly finds an optimal policy with a function form.

II. Related Work

The first thread of work that we follow is the MDP or POMDP based service assistance agents. In [2], the authors propose a hand-washing assistance agent based on an MDP formulation (and is later generalized into a POMDP [3]). Different from the complex MDP formulated, the MDP formulated in this work has finite number of states and actions and thus cannot be applied to our problem. A more general framework of service assistance agent is proposed in [4]. In this work, the assisted tasks are not specified, and thus cannot be directly applied to solve the domain specific problem of cooking assistance.

We follow the second thread of work of solving large scale MDPs. There have been a great amount of works on solving large scale MDPs. One category of methods focuses on tabular representations of the value and policy functions. These tabular-based methods include tabular tabular Q-learning [5], prioritized sweeping [6], Monte Carlo Tree Search (MCTS) [7] and UCT (UCB applied to trees) [8]). However, the state space of our formulated MDP is continuous, and thus cannot be represented with a tabular form.

Another category of methods use approximated functions to represent the value functions.

Traditional function approximation methods [9], [10], [11] perform action selection by approximating state-action values. Due to the continuous action spaces, the action selection process becomes rather inefficient with these traditional value function approximation methods. As a result, we adopt a policy gradient method [12], [13] which directly represents the policy as a parameterized function of states.

III. Formulation

A. Cooking System Formulation

Formally, we consider a cooking task T which consists of a set of sequential subtasks $\{T_1, \dots, T_n\}$. We assume that a subtask $T_i (i \geq 1)$ can be triggered only when its preceding task T_{i-1} is completed. This is intuitive in a cooking task. Take Spaghetti as an example, the subtask of cooking Spaghetti can be triggered only when the subtask of boiling water is completed.

A cooking assistance agent is able to remind the patient when the patient is performing each subtask. The reminder can be triggered at time t_i during each task, where finish of the last subtask T_{i-1} is set by default as $t_i = 0$. Generally, by triggering a reminder earlier, the patient would have a higher utility by completing the subtask with a shorter time, but he/she will also experience a higher level of frustration. We denote the payoff of the patient completing a subtask $T_i \in T$ as $v_i(t_i)$, which is assumed to be a decreasing function w.r.t. time t_i. We denote the frustration loss of the patient being reminded at time t_i while completing a subtask T_i as $c_i(t_i)$, which is assumed to be a decreasing function of time.

B. MDP of Cooking Assistance Agent

We formulate the planning problem of the assistance agent as an MDP. Formally, the state of the MDP is a vector $\mathbf{s} = \langle s_1, \dots, s_n, s_{n+1} \rangle$ where each $s_i = \langle s_1, \dots, s_n \rangle$ represents the time it takes for the patient to accomplish each task T_i, and $s_{n+1} = i$ indicates the subtask T_i the patient is starting to perform. If a subtask T_i has already been completed (i.e., $s_{n+1} > i$), then s_i means the actual task completion time. Otherwise, s_i is the estimated task completion time. Intuitively, the system state reveals the mentality status of the patient, i.e., if the patient is able to complete the subtasks within a shorter time, it is believed that the patient is in a better status, and tends to be able to complete the remaining subtasks faster. We denote the estimated task completion time for each subtask T_i as a probability distribution function of state: $f(t_i, \mathrm{s})$, where t_i^{max} is the longest time a patient can complete subtask T_i. It is required that

$$\int_{t_i=0}^{t_i^{max}} f(t_i, \mathrm{s}) = 1. \qquad (1)$$

The action of the assistance agent is to decide, for each subtask T_i, the time a_i a reminder is released to the patient during the subtask, given the current state s. After the reminder for subtask T_i is released, we assume that the patient is able to complete the subtask within a fixed time t_i^0. Note that the actions taken after a subtask T_i is completed will not affect the task completion time of the previous subtasks anymore. As a result, taken action a_i, the expected utility of competing a subtask T_i (i.e., the immediate reward) is denoted as

$$r_i(a_i) = \int_{t_i=0}^{a_i} v_i(t_i) f(t_i, \mathrm{s}) dt_i + \int_{t_i=a_i}^{t_i^{max}} v_i(a_i + t_i^0) f(t_i, \mathrm{s}) dt_i. \qquad (2)$$

The first integration on the right hand side denotes the expected utility of the patient if he/she completes the subtask T_i before the reminder is released, while the second integration denotes the other case. The long term reward or value function is represented as a sum of all the immediate reward of the tasks completed after subtask T_i:

$$v(\mathrm{s}, a_i) = \sum_{j=i}^{n} r_j(a_j). \qquad (3)$$

A policy $\pi(a_i|\mathrm{s})$ is a function which specifies the conditional probability of taking an action a_i, given a certain state s. The optimal policy maximizes the value function in Eq. (3):

$$\pi^*(a_i|\mathrm{s}) = \arg\max_{\pi} v(\mathrm{s}, a_i). \qquad (4)$$

The state transition function is the probability of a state s transiting into state s', given certain action a_i. In the next state, since only s_i is changed, it is equivalent to the probability of the patient using s_i time to complete the subtask T_i:

$$p(\mathrm{s}'|\mathrm{s}, a_i) = p(s_i|\mathrm{s}, a_i). \qquad (5)$$

C. State Transition Function

As we can see, the state transition function depends on the probability distribution function $f(t_i, \mathrm{s})$. While our formulation above is general to all types of $f(t_i, \mathrm{s})$, in this subsection, we study a specific form, i.e., a normal distribution. More specifically, we consider a probability distribution function of the completion time t_i of subtask T_i, given state s:

$$f(t_i, \mathrm{s}) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\{-\frac{(t_i - \mu_i)^2}{\sigma_i^2}\}, \qquad (6)$$

where $\mu_i = \mu_i(\mathrm{s})$ and σ_i are respectively the mean and standard deviation of the distribution. We assume

$$\mu_i = \bar{\mu}_i \frac{\sum_{j=1}^{i-1} s_j}{\sum_{j=1}^{i-1} \bar{\mu}_j}, \qquad (7)$$

where $\bar{\mu}_i$ is the average task completion time of subtask T_i obtained from historical statistics. We also assume σ_i can be obtained from historical statistics and does not change over time. Intuitively, this indicates that if the task completion time of the previous subtasks is long, then the task completion time of subtask T_i would also be long. As a result, we have

$$f(s_i|\mathrm{s}, a_i) = \begin{cases} f(t_i, \mathrm{s}), & s_i \le a_i; \\ \int_{a_i}^{\infty} f(t_i, \mathrm{s}) dt_i, & s_i = a_i + t_i^0. \end{cases} \quad (8)$$

IV. Analysis and Proposed Methods

As we can see, the formulated MDP is a finite horizon MDP with continuous action and state spaces. Solving this formulated MDP is very challenging due to two major reasons. First, the formulated MDP has very large scale action and state spaces. For such large scale MDP, it is difficult to even represent the value functions with a tabular form. As a result, we use an approximated function form to represent the value function. In general, function approximation [14], [15], [16] selects a function among a well-defined class that closely matches ("approximate") a target function in a task specific way. The novelty of function approximation is that the approximate value function $v(\mathrm{s}, \theta)$ is represented not as a table but as a parameterized functional form with weight vector θ. $v(\mathrm{s}, \theta)$ might be a linear function of the features of the state, with θ as the weight vector of the features. More generally, $v(\mathrm{s}, \theta)$ might be the function computed by a multilayer artificial neural network, with θ the vector of connection weights in all the layers. By adjusting the weights, any of a wide range of different functions can be implemented by the network.

Even with function approximation, the search spaces of action and state are still infinitely large, which makes it infeasible for traditional search methods such as Monte Carlo Tree Search (MCTS) [7], UCT (UCB applied to trees) [8] or function approximation methods [9], [10], [11] that perform action selection by approximating state-action values cannot be applied to our problem. As a result, we propose a policy gradient method in reinforcement learning to solve the problem.

The idea of policy gradient method is to approximate the policy function with a parameterized function $\pi(\mathrm{a}|\mathrm{s}, \vartheta)$, and update the parameter with the stochastic gradient descent method. Similar to the value function, the form of the approximated policy function can either be linear to the features of the state or an artificial neural network. More specifically, PG-β incorporates an actor-critic architecture, where “actor” is the learned policy function which performs action selection, and “critic” refers to the learned value function which is a performance measure of the current policy function. As shown in Algorithm 1, the input of the policy gradient method includes the state transition function, a set of parameterized value functions $v_i(\mathrm{s}, \theta_i), \forall i = 1, \ldots, n$ with respect to parameter $\theta_i, \forall i = 1, \ldots, n$ and a set of parameterized policy functions $\pi_i(a_i|\mathrm{s}, \vartheta_i), \forall i = 1, \ldots, n$ with respect to parameter $\vartheta_i, \forall i = 1, \ldots, n$. More specifically, we adopt the most commonly used form of value function, where the value function is linear with respect to the feature functions $v_i(\mathrm{s}, \theta_i)$, i.e.,

$$v_i(\mathrm{s}, \theta_i) = \sum_j \theta_{ij} \phi_j.$$

For the choice of a proper policy function, a key aspect is to trade-off the balance between exploitation (of a good policy) and exploration (of potential better policy). To do this, we use a normal distribution function to represent the policy function, instead of using a deterministic policy function. Formally, we denote the policy function as

$$\pi_i(a_i|\mathrm{s}, \vartheta_i^\sigma, \vartheta_i^\mu) = \frac{1}{\sqrt{2\pi\sigma_i'^2}} \exp\{-\frac{(a_i - \mu_i')^2}{\sigma_i'^2}\},$$

where σ' and μ' are respectively the standard deviation and mean value of the normal distribution used to sample the action a_i, which are linear functions with respect to the feature functions $\phi(\mathrm{s})$ with the parameters ϑ^σ and ϑ^μ.

```
Algorithm 1: Policy Gradient
1 Initialize $\vartheta \leftarrow \vartheta_0$, $\theta \leftarrow \theta_0$;
2 repeat
3     Generate an episode $\mathrm{s}^1, \mathrm{a}^1, R^1, \ldots, \mathrm{s}^n, \mathrm{a}^n, R^H$;
4     $Q \leftarrow \sum_{i=1}^{n} r_i$;
5     $\delta \leftarrow Q - \hat{v}(\mathrm{s}, \mathrm{a}, \theta)$
6     $\vartheta \leftarrow \vartheta + \beta\delta\nabla_{\vartheta}\hat{v}(\mathrm{s}, \vartheta)$;
7     $\theta \leftarrow \theta + \beta'\delta\nabla_{\theta}\log\pi(\mathrm{a}|\mathrm{s}, \theta)$
8 until forever
9 return $\theta$;
```

The algorithm starts with an initialization of the parameters ϑ and θ. It then enters the repeat loop (Lines 2-8). In the repeat loop, it first simulates an episode, which contains a series of state-action pairs in the planning horizon H. The states are generated following the state transition function Eq.(5), the actions are selected following the policy function $\pi(\mathrm{a}|\mathrm{s}, \boldsymbol{\theta})$, while the immediate rewards are computed with Eq.(2). After an episode is simulated, the algorithm then updates the parameters (Lines 4-7) with stochastic gradient descent method. Q denotes the sum of rewards of completing all the subtasks obtained from the simulated episode, which reveals the "real" value obtained by the current policy, while $\hat{v}^t(\mathrm{s}^t, \boldsymbol{\vartheta}^t)$ is the "estimated" sum of rewards approximated by the parameterized value function $v^t(\mathrm{s}^t, \boldsymbol{\vartheta}^t)$. Consequently, δ denotes the difference of these two terms. In Lines 6-7, policy gradient method updates the parameters with the specified gradient methods.

V. Conclusion

With the rapid growth of population affected with Alzheimer's Disease, human society is faced with a fast growing cost of providing home care to patients affected by Alzheimer's. In order to contribute to saving this home care cost, we propose a cooking assistance agent to help the patients during the daily cooking. Our agent formulates the cooking system as an MDP with continuous state and action spaces. To cope with the large scale state and action space, we adopt a category of reinforcement learning methods called “policy gradient” method.

In the future, we plan to test our approach with both simulation-based and real-world data. Moreover, we will generalize our formulated MDP into POMDP, where the activities of the patients cannot be fully observed and evaluated.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its IDM Futures Funding Initiative.

References

[1] A Place for Mom Inc. 2016 alzheimers statistics. 2017. http://www.alzheimers.net/resources/alzheimers-statistics/.

[2] Jennifer Boger, Jesse Hoey, Pascal Poupart, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A planning system based on markov decision processes to guide people with dementia through activities of daily living. IEEE Transactions on Information Technology in Biomedicine, 10(2):323–333, 2006.

[3] Jesse Hoey, Pascal Poupart, Axel von Bertoldi, Tammy Craig, Craig Boutilier, and Alex Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. Computer Vision and Image Understanding, 114(5):503–519, 2010.

[4] Jesse Hoey, Craig Boutilier, Pascal Poupart, Patrick Olivier, Andrew Monk, and Alex Mihailidis. People, sensors, decisions: Customizable and adaptive technologies for assistance in healthcare. ACM Transactions on Interactive Intelligent Systems (TiiS), 2(4):20, 2012.

[5] Christopher John Cornish Hellaby Watkins. Learning from Delayed Rewards. PhD thesis, University of Cambridge England, 1989.

[6] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning, 13(1):103–130, 1993.

[7] R´emi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In International Conference on Computers and Games, pages 72–83. Springer, 2006.

[8] Levente Kocsis and Csaba Szepesv´ari. Bandit based monte-carlo planning. In European conference on machine learning, pages 282–293. Springer, 2006.

[9] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In ICML, pages 417–424, 2001.

[10] Hamid R Maei, Csaba Szepesv´ari, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In ICML, pages 719–726, 2010.

[11] Barry D Nichols and Dimitris C Dracopoulos. Application of newton's method to action selection in continuous state-and action-space reinforcement learning. ESANN, 2014.

[12] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

[13] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In NIPS, volume 99, pages 1057–1063, 1999.

[14] Leemon Baird et al. Residual algorithms: Reinforcement learning with function approximation. In Proceedings of the twelfth international conference on machine learning, pages 30–37, 1995.

[15] Geoffrey J Gordon. Stable function approximation in dynamic programming. In Proceedings of the twelfth international conference on machine learning, pages 261–268, 1995.

[16] J Geoffrey. Reinforcement learning with function approximation converges to a region. 2001.

Haipeng Chen received the B.S. degree from the School of Physics, University of Science and Technology of China, Hefei, China, in 2013. He is currently pursuing the Ph.D. degree with the Joint NTU-UBC Research Center of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore. His research interests are in the area of game theory, multi-agent systems, artificial intelligence, machine learning, and cloud computing.

Yeng Chai Soh (M’87–SM’06) received the B.Eng. degree (Hons.) in electrical and electronic engineering from the University of Canterbury, New Zealand, and the Ph.D. degree in electrical engineering from the University of Newcastle, Australia. He has served as the Head of the Control and Instrumentation Division, the Associate Dean of Research and Graduate Studies, and the Associate Dean of Research with the College of Engineering. He joined the Nanyang Technological University, Singapore, where he is currently a Professor with the School of Electrical and Electronic Engineering. He has authored over 260 refereed journal papers in these areas. His most recent research projects and activities are in sensor networks, sensor fusion, distributed control and optimization, and control and optimization of ACMV systems. His research interests have been in robust control and applications, robust estimation and filtering, optical signal processing, and energy efficient systems. He has served as panel members of several national grants and scholarships evaluation and awards committees.