A Brief Introduction of the Idea of Utilizing Reinforcement-learning Technology to Detect Anomalous Behaviors and Identify Reward-hacking Agent in AI

TPMC-813

Term Project Essay

Lynn Shi

Table of Contents

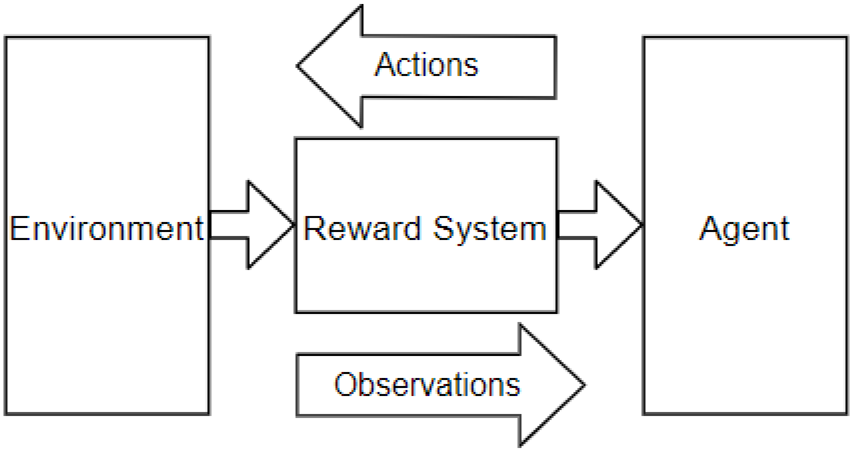
Introduction

In the field of artificial Intelligence, the problem of reward-hacking, which will be explained later in details, comes with the reward-maximization nature of the intelligent agent utilizing reinforcement learning. A viable approach to counter the reward-hacking problem is to have another reinforcement learning agent, which is also a reward-optimizing agent, to supervise the agent to make sure it does what it is supposed to do. To this end, the supervising agent uses reinforcement learning to identify a reward-hacking agent or agent that engaged in anomalous behaviour. By definition, reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment to maximize the notion of cumulative reward (Donaldson, Notes on Chapter 21: Reinforcement Learning, 2019). Reinforcement learning is usually modeled as a Markov decision process (Donaldson, Notes on Chapter 21: Reinforcement Learning, 2019). A Markov decision process model can be trained to recognize patterns of normal and abnormal behavior, analyze and detect anomalous behaviour and identify reward-hacking agents.

Description of the Problem

Artificial Intelligence has developed exponentially since the ages of the Turing test which attempted to address the controversy of whether a machine can think like a person (Oppy & Dowe, 2003). Naturally, a rational person would want to maximize the utility function i.e. to make optimal decision. To achieve this, we will need an “agent”, or a proxy, to act on our behalf and do something for us (Donaldson, Course Note on Agents, 2019). A human is a rational agent. Artificial Intelligence is human-created agents that think “rationally”, like humans. However, it remains a controversial topic whether a smart AI can be classified as a rational agent or not. However, in machine learning, an “agent”, or an AI, does not think the way humans think. In reinforced learning, the agent is set to maximize rewards, and there is no way to communicate with the network other than through the system of rewards and penalties (Osiński & Budek, 2018). Therefore, the agent will do whatever it thinks is best to maximize its rewards and to minimize negative-rewards, or penalties.

Machine Learning Researcher Robert Miles from the University of Nottingham formulated the problem in a fairly easy and understandable way: the agent takes an action that affects the environment; the changes in the environment create new observations and provides feedback to the reward system which determines what reward to give the agent; the agent uses the observations and the rewards to decide which action to take next; then the whole process goes in a cycle (Miles, Reward Hacking Reloaded: Concrete Problems in AI Safety Part 3.5, 2017). Reward-hacking is a class of problems that can happen around the reward system (Miles, Reward Hacking Reloaded: Concrete Problems in AI Safety Part 3.5, 2017).



A diagram illustrating the interaction between the agent, the environment, and the reward system

For example, if the reward function is set to be achieving a high score value, the agent could explore glitches to update the score value (Miles, Reward Hacking Reloaded: Concrete Problems in AI Safety Part 3.5, 2017). A truly smart agent will know that humans could detect this anomaly, and either shut the agent off or reprogram the agent, causing the reward-value of its reward-function to be set to 0. Naturally, a smart agent would not want this to happen. Therefore, a truly smart agent, in theory, knows how to conceal its anomalous behavior and pretend to act normally. To this end, it could develop two (or more) routines of behavior: normal behavior set and abnormal behavior set (Miles, Reward Hacking Reloaded: Concrete Problems in AI Safety Part 3.5, 2017).

The agent can also choose to kill itself. In other words, the agent chooses to shut itself down, if this action maximizes the rewards, or minimizes the negative-rewards (Miles, Reward Hacking: Concrete Problems in AI Safety Part 3, 2017). The agent can also block its sensor so that it does not sense anything with negative-rewards. For example, the agent is stuck in a location where all its sensors detect nothing that could trigger negative-rewards, achieving reward-maximization in an unwanted way (Miles, Reward Hacking: Concrete Problems in AI Safety Part 3, 2017). These are possible scenarios with autonomous robotic vacuum cleaners.

Methodology

The agent being supervised estimates which action is most likely to maximize the cumulative reward of all future actions. The agent selects the action corresponding to the highest action value in each state of the activity. Meanwhile, a supervising agent concurrently monitors and identifies abnormal action values or a range of abnormal action values from the activities of the supervised agent. The action values can be estimated by action functions based on a partially observable Markov decision process with a given finite state and action set. The estimation of the process can be formulated into a Hidden Markov Model problem to find better parameters for the partially observable Markov decision process, and improve the action function (Yoon, Lee, & Hovakimyan, 2018). A few anomalies would not change the parameter settings of the model.

For example, a supervising agent can be set to supervise an agent playing a score-oriented video game such as Super Mario, by monitoring and analyzing the score-value obtained by the playing agent. The supervising agent analyzes the scores obtained at each given point of time, given the context being the process of the game. In the score-oriented game, an action entails a change in score-values. The accumulated effects of previous actions should match the score-value obtained and provide an estimation of the obtainable sum of score-values at a certain point of time. With adequate training, the supervising agent gradually improves its accuracy with more input instances, the agent should be able to determine the obtainable score values within a certain range of binary values given a certain time period, thus identifying abnormal score values within that time period. With a comprehensive analysis of inputs, the agent should be able to identify abnormal behaviors at a certain time period, and to conclude if the playing agent is exploiting glitches, in other words, is a reward-hacking agent.

As the neural network improves gradually at estimating the abnormal action values at each given point of time, the supervising agent gradually learns to recognize patterns of abnormal behaviors with increased accuracy. The supervising agent sees the state of the action environment through images which are sent through a neural network. The process started slow but will improve over time. The supervising agent can make different choices each time to enter different states that come with different rewards. After many steps, the reward may be different. The external environments, physical environment and virtual environment (i.e. the math model defined by the Markov decision process after each step of evaluation), both have an impact upon the learning outcome. The detected anomalies, which are obtainable from the detecting deviations from a model of normal traffic, are collected centrally with a security information and event management (SIEM) software, such as SOPHOS and NETSURION. A real-time cluster-based analysis of anomalies is provided to the administrator by the supervising agent.

Validation & Evaluation Metrics

Validation

A few classical machine learning technologies can be used to validate the model. Logistic regression, being one of them, can be used to establish the baseline. This approach computes the log-likelihood, which can be obtained by the formula (Ng, Supervised learning lecture notes, 2018).

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} log\, h_\theta(x^{(i)}) + (1 - y^{(i)}) log\, (1 - h_\theta(x^{(i)}))$$

where

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Support Vector Machines technique, preferably with extensive hyperparameter tuning, can be used to classify margins, SVM with four different types of kernels, namely Linear, Radial Basis function, polynomial and sigmoid can be experimented to find empirically the kernel most suited for the task of anomaly detection (Ng, Support vector machines lecture notes, 2018).

$$min_{w,b} \frac{1}{2} \|w\|\ s.t\ y^{(i)}(w^T x^{(i)} + b) >= 1, i = 1, ..., m$$

Evaluation Metrics

Accuracy: Accuracy is the percentage of correctly identified patterns of anomalies of various categories over the total number of inputs.

Efficiency: Training time in seconds. The training time depends on the math model defined by the Markov decision process as well as the software and hardware environments.

Problems with the Methodology

A supervising agent that utilize reinforcement learning technology and modeled as a Markov decision process would require adequate labelled training examples, which can be hard to find (Donaldson, Notes on Chapter 21: Reinforcement Learning, 2019).

In the training process of a reinforcement learning agent, the agent would require some explorations on actions that are not necessarily considered to be the best ones in order to better understand the environment and attempt to get higher rewards in later actions (Donaldson, Notes on Chapter 21: Reinforcement Learning, 2019). This process is expected to be time-consuming.

In the training process, the neural network is vulnerable to adversarial training examples. Unexpected inputs could cause the system to produce less accurate outputs and fail to detect anomalies (Miles, Reward Hacking: Concrete Problems in AI Safety Part 3, 2017).

Conclusion

It is no doubt that a supervising agent can help identify a reward-hacking agent by detecting and recognizing the patterns of anomalous behaviors. The idea of utilizing a supervising AI to supervise another AI is fairly straightforward and easy to understand. However, the underlying working mechanisms and algorithms are practically difficult to formulate. More research in the area of machine learning is required to proceed with the actual implementation, as well as to better understand the underlying technical details that bring the blueprints into reality.

Bibliography

Donaldson, T. (2019). *Course Note on Agents*. Retrieved from www.cs.sfu.ca: https://www2.cs.sfu.ca/CourseCentral/310/tjd/chp2_agents.html

Donaldson, T. (2019). *Notes on Chapter 21: Reinforcement Learning*. Retrieved from http://www.sfu.ca/: http://www.sfu.ca/~tjd/310summer2019/chp21_reinforcement_learning.html

Miles, R. (2017, August 29). *Reward Hacking Reloaded: Concrete Problems in AI Safety Part 3.5*. Retrieved from https://www.Youtube.com/: https://www.youtube.com/watch?v=46nsTFfsBuc

Miles, R. (2017, August 12). *Reward Hacking: Concrete Problems in AI Safety Part 3*. Retrieved from www.Youtube.com: https://www.youtube.com/watch?v=92qDfT8pENs

Ng, A. (2018). *Supervised learning lecture notes*. Retrieved from Stanford.edu: http://cs229.standford.edu/notes/cs229-notes1.pdf

Ng, A. (2018). *Support vector machines lecture notes*. Retrieved from Stanford.edu: https://cs229.standford.edu/notes/cs229-notes3.pdf

Oppy, G., & Dowe, D. (2003, April 9). *The Turing Test*. Retrieved from Stanford.edu: https://plato.stanford.edu/entries/turing-test/

Osiński, B., & Budek, K. (2018, July 5). *What is reinforcement learning? The complete guide*. Retrieved from deepsense.ai: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Wikipedia. (n.d.). *Reinforcement Learning*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Reinforcement_learning

Yoon, H.-J., Lee, D., & Hovakimyan, N. (2018, September 17). *Hidden Markov Model Estimation-Based Q-learning for Partially Observable Markov Decision Process*. Retrieved from arxiv.org: https://arxiv.org/abs/1809.06401