Machine Learning Nanodegree
Capstone Report

David A. Robles

January 31, 2017

1 Definition

1.1 Project Overview

Content.

1.2 Problem Statement

In this project, we will use reinforcement learning with deep learning to make an agent learn to play the game of Connect 4[1] by playing games against itself. In other words, using the formalism used by Mitchell (1997) to define a machine learning problem:

- **Task:** Playing Connect 4.
- **Performance:** Percent of games won against other agents, and accuracy of the predictions on a Connect 4 dataset.
- **Experience:** Games played against itself.
- **Target function:** $Q^{\pi} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where $\mathcal{S}$ is the set of *states* (board positions) and $\mathcal{A}$ is the set of *actions* (moves), and $\mathbb{R}$ represents the value of being in a state $s \in \mathcal{S}$, applying a action $a \in \mathcal{A}$, and following policy π thereafter.
- **Target function representation:** Deep neural network.

Therefore, I seek to build a Q-learning agent trained via a deep convolutional neural network to approximate the optimal action-value function:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{1}$$

which is the maximum sum of rewards achievable by a behaviour policy π.

1.3 Metrics

- **Winning percentage.** This metric consists in playing a high number of games (e.g. 100,000) against another agent (e.g. a random agent), and calculating the average of games won by the agent that uses the learned value function.
- **Prediction accuracy.** The learned value function will be used to predict the game-theoretic outcomes (win, loss or draw) of the board positions in the Connect 4 Data Set.

[1] https://en.wikipedia.org/wiki/Connect_Four

2 Analysis

2.1 Data Exploration

Content.

2.2 Exploratory Visualization

Content.

2.3 Algorithms and Techniques

2.3.1 Q-learning

One of the most basic and popular methods to estimate action-value functions is the *Q-learning* algorithm (Watkins, 1989). It is model-free online off-policy algorithm, whose main strength is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Q-learning works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. The update rule uses action-values and a built-in max-operator over the action-values of the next state in order to update $Q(s_t, a_t)$ as follows,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2}$$

The agent makes a step in the environment from state s_t to s_{t+1} using action a_t while receiving reward r_t. The update takes place on the action-value a_t in the state s_t from which this action was executed.

Q-learning is exploration-intensive, which means that it will converge to the optimal policy regardless of the exploration policy being followed, under the assumption that each state-action pair is visited an infinite number of times, and the learning parameter α is decreased appropriately (Watkins and Peter, 1992).

2.4 Benchmark

Content.

3 Methodology

3.1 Data Preprocessing

Content.

3.2 Implementation

Content.

3.3 Refinement

Content.

4 Results

4.1 Model Evaluation and Validation

Content.

4.2 Justification

Content.

5 Conclusion

5.1 Free-Form Visualization

Content.

5.2 Reflection

Content.

5.3 Improvement

Content.

References

T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

Christopher J.C.H. Watkins and Dayan Peter. Q-learning. *Machine Learning*, 8:279–292, 1992.

C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.