Q-Learning: Reducing the State Space with Neural Networks

Boris Repasky, Suraj Narayanan, Sina Eghbal

{u5844485, u5881495, u5544352}@anu.edu.au
Australian National University, Canberra, Australia

Motivation

Q-learning entails learning the value of every state-action pair.

1. Number of states in even a small problem is very large[1]
2. Want to extract features automatically, given any problem
3. Want to generalise to previously unseen observations

Contribution: Incorporating an autoencoder into Q-learning in order to compress the state space of a Q-learning problem.

The Problem

Under simplifying assumptions, considering only the locations of the ghosts(G) and of Pacman (keeping the walls (W) and food (F) fixed), the number of possible configurations is

$$2^F \frac{(N \times M - W)!}{G!(N \times M - W - G - 1)!} \tag{1}$$

which for a 27×28 level with 4 Ghosts, 462 walls and 229 food pellets is already over 8×10^{81}.
Adding other game components the number of possible states quickly becomes intractable. Storing Q-values for every state-action pair is infeasible.

General Framework

General Reinforcement Learning(RL): for each time step t, an *agent* observes a *state* $s_t \in S$ and receives a *reward* r_t, then selects an action $a_t \in A(s_t)$.

- **Task**: develop a *policy* $\pi : S \to A$ that maximises expected future reward $R_t = \sum_{k=0}^{\infty} \gamma^t r_{t+k+1}$ where $\gamma \in [0, 1]$

Q-Learning: An off-policy, model-free, temporal-difference method to solve the RL problem.

- **Central Idea**: Approximate the optimal action-value function Q^* directly using the following update formula.[3]

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha\Big(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\Big)$$

Can perform update independent of the policy being followed. Greatly simplifies the RL problem.

Autoencoder(AC): Can be defined as a pair of functions, an *encoder* $\phi_{ENC} : S \to Z$ and a *decoder* $\psi_{DEC} : Z \to S$.
We take S to be the set of possible *game states* of a Pacman level, then Z is the *encoded feature space*.

Simplest AC architecture: Minimise the reconstruction error $\mathcal{L}(s, (\phi \circ \psi)(s))$ subject to the constraint that $dim(Z) < dim(S)$.

- Ensures that we do not learn the identity function.
- Can view $\phi_{ENC}(s) = z$ as a compressed representation of the input s.
- AC is an *unsupervised learning* technique

We further simplify by defining a *projection* $\sigma : Z \to \big[\mathbb{N} \cap [0, 350]\big]^{dim(Z)}$

- Reduces cardinality of final encoding to $350 \times dim(Z)$.
- Discretises encoding, groups together similar states.

AC pre-trained with simplified game states using *Binary cross entropy* and *ADADELTA*.[4]

Three Q-learning Models

Vanilla Q-learning: (Q_V)
The standard Q-learning algorithm, a separate Q-value is stored for each state-action pair.

Q-learning with Autoencoder: (Q_{AC})
Simple Q-learning combined with an autoencoder layer, at each time step t, state s_t is transformed via $\sigma \circ \phi_{ENC}(s_t) = z'_t$ and the Q-values are updated by

$$Q_{t+1}(z'_t, a_t) = Q_t(z'_t, a_t) + \alpha\Big(r_{t+1}(s_t, a_t) + \gamma \text{max}_a Q_t(z'_{t+1}, a) - Q_t(z'_t, a_t)\Big)$$

Approximate Q-learning[2]: (Q_F)
Q-function is approximated as a linear combination of hand selected features f_i as follows

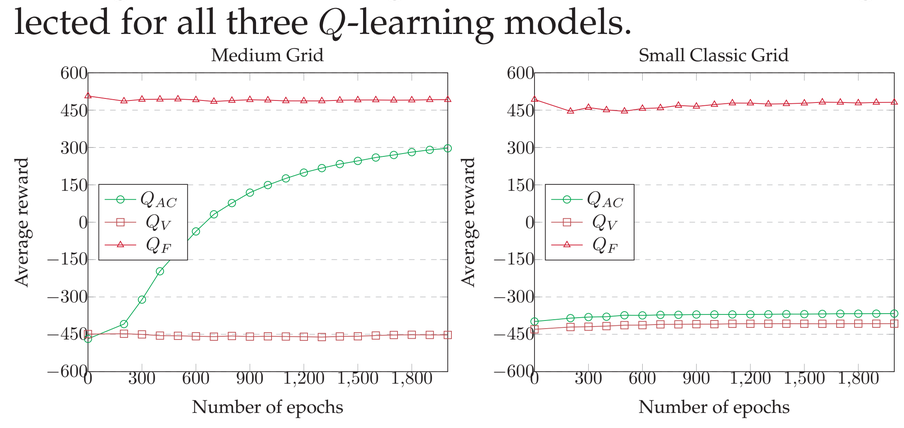
$$Q(s, a) = \sum_i^n f_i(s, a) w_i.$$

Then weights w_i are updated in each time step t according to

$$w_{i,t+1} \leftarrow w_{i,t} + \alpha\big(r_{t+1}(s_t, a_t) + \gamma \text{max}_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t)\big)$$

Experimental Evaluation

- Q_V, Q_{AC}, and Q_F tested and compared on four different PacMan levels of varying complexity.
- Average reward each agent received after 'n' training iterations collected for all three Q-learning models.



- AC outputs shows large error in location of Pacman and ghosts, with the walls preserved the most faithfully.

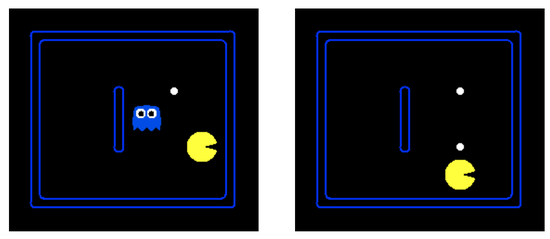


Fig. 1. Pacman actual level layout(left) and decoded output(right).

- AC devoting too many resources to replicating the level border. AC retrained without borders improved results.

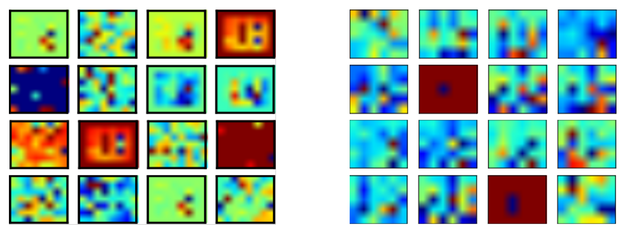


Fig. 2. AC weights visualisation, AC trained with borders (left) compared to AC trained without borders (right)

- Q_V-learning performs well only on the smallest of grids, Q_{AC} performs well on medium grid but still learns Q-values slowly.

Conclusions & Future Work

- Q_{AC} outperforms Q_V in all test scenarios, however cannot achieve performance of Q_F, especially in larger more complex levels.
- Suggests that the autoencoder is learning useful features but learnt features are not competitive with human picked features.
- Found autoencoder has a bias to preserving walls and empty spaces, failing to encode the ghost positions or the agents position. This increases the stochasticity of the environment.
- Improvements gained by removing the borders from input states, neurons aren't wasted replicating unnecessary information.
- **Extensions**: Use an error function that penalises incorrect Pacman and ghost positions more heavily.
- Use features extracted by a trained autoencoder together with a Q-learning function approximation method.

[1] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* (1996), pp. 237–285.

[2] Miquel Ramirez. *COMP3620/6320 Assignment 4*. [Online; accessed 22-May-2016]. 2016. URL: https://gitlab.cecs.anu.edu.au/u5680374/comp3620-2016-assignment-4.

[3] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. Vol. 135. MIT Press Cambridge, 1998.

[4] Matthew D Zeiler. "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).