State-Space Reduction in Deep Q-Networks

Statistics 234 | Professor Susan Murphy | Spring 2018

Michael Ge - michaelge@college.harvard.edu | Richard Ouyang - rouyang@college.harvard.edu

ABSTRACT

Deep convolutional neural networks have become a popular approach to estimating Q-value functions in reinforcement learning problems. These deep Q-networks take in entire images as network inputs, often resulting in large state spaces and long learning times. In this paper, we explore the use of principal component analysis to reduce the state space of network inputs. After testing multiple network configurations, we determine that a reduction in uninformative state features through PCA helps improve the performance of deep reinforcement learning.

BACKGROUND

We briefly discuss the application of deep neural networks to reinforcement learning. The most common use of deep neural networks is to model the Q-function. These **deep Q-networks (DQNs)** take in a transformed image input s, pass it through multiple layers, and return a vector consisting of the estimated Q(s, a) for all possible actions a.

Current state-of-the-art models (Mnih et al.) use **deep convolutional neural networks (CNNs)** as models for the Q-function; these DQNs capitalize on the structure of image data – for example, correlations between nearby pixels – to make Q estimates. Unfortunately, DQNs typically require a large number of parameters, thus consuming large amounts of computational resources, both in terms of time and space.

Double DQNs (DDQNs) (Van Hasselt et al.) are very similar to regular DQNs. However, instead of using the same network to obtain the target value for minibatch updates as in a regular DQN, the DDQN randomly selects one of the two networks to update and uses the other network to obtain the target value.

Principal component analysis (PCA) is a dimensionality reduction technique that linearly projects the original features into a smaller dimension. PCA chooses the axes with the most variance to construct this smaller space. The resulting features have zero correlation with one another. In interpreting images, PCA keeps transformations of the most variable pixels, eliminating the information contained in pixels with very little variance.

OpenAI Gym offers a set of many standard games on which to test reinforcement learning algorithms. We tested our algorithms, which learn exclusively on pixel images, on three games: CartPole, where the agent attempts to balance a pole on top of a cart; Acrobot, where the agent swings a double pendulum above a certain height; and MountainCar, where the agent tries to drive an underpowered car up a steep hill.

PyTorch is a Python framework for constructing neural network architectures. Although PyTorch is still in its early stages of development, it has several advantages compared to other neural network libraries such as TensorFlow and Keras. Namely, improvements include a clean extension of the common Python package NumPy, deep integration with Python, and efficient memory usage.

NETWORK STRUCTURE

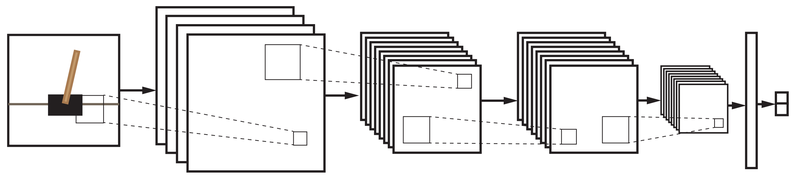
In our experiments, we used four different types of models, each of which includes the original and double network variants:

(D)DQCNN: The original model and variants (Mnih et al., Van Hasselt et al.). These models convolve images using no extra state-space reduction techniques like PCA.

(D)DQN-PCA: These models use a database of 1000 states as training data for PCA and project the pixel features of new states onto the subspace that captures 99% of the variance. The result is a one-dimensional vector with varying lengths based on the game (100 to 500 features). This vector is passed into a conventional (not convolutional) deep neural network to predict the Q-value.

(D)DQCNN-PCA: These models perform PCA as above, but invert the PCA transformation in order to convert each image back to its original space. The result is again an 80 x 80 image, which is then input into a model with the same network structure as the original (D)DQN.

(D)DQCNN-PCA-Mini: Out of curiosity, we included one additional model, which takes the one-dimensional vector obtained from the PCA transformation and reshapes it into a square (zero padding if necessary). We then convolved the resulting "image" through a smaller neural network to predict the Q-value for each state. Although PCA strips correlation between nearby pixels, we were interested in investigating whether the reduced dimensionality preserved some structure between nearby values.



Deep Convolutional Neural Network Example

HYPERPARAMETERS

Here is a selection of hyperparameters that we varied:

Model: One of the Q-function models.

Number of minibatch training steps: The length of time the model should be trained, expressed in terms of the number of minibatch training steps.

Target update: The number of trains between each target network update. Each update involves setting the target network parameters to the main network parameters.

Learning rate: The (initial) learning rate used by the optimizer. A higher rate results in a model that learns faster but may diverge.

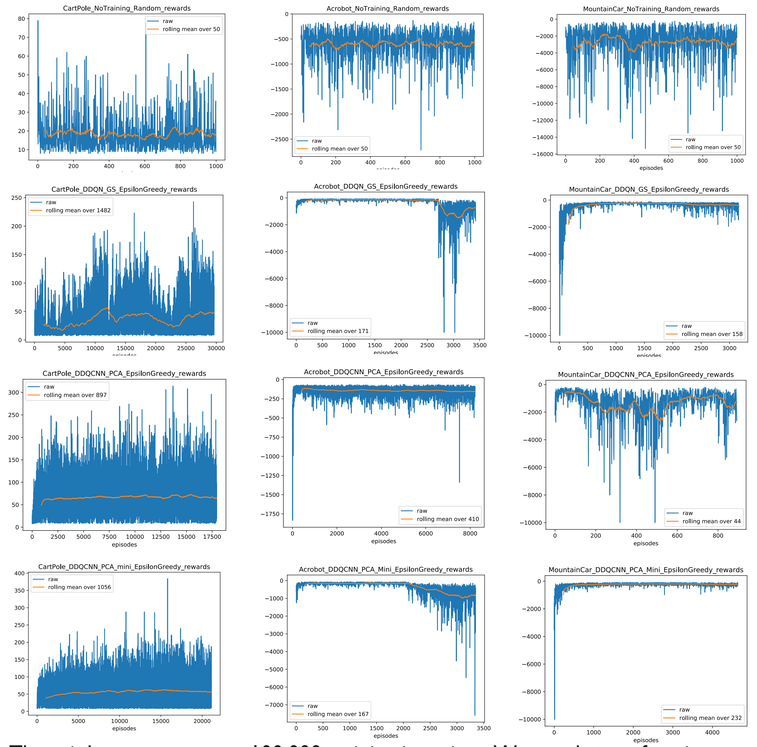
Learning rate annealing: Whether the learning rate is annealed to a smaller value. Learning rate annealing helps decrease the likelihood of policy divergence.

Batch size: The minibatch size, or the number of transitions used in each training update. The two values tested were 32 and 128.

Loss function: The loss function used in the model. Either Huber or MSE loss.

Regularization: The weight decay in the optimizer. This value typically ranges from 0 to 1. This value is used to prevent overfitting in our networks.

RESULTS



These jobs were run over 100,000 training iterations. We see that performing PCA on the input states seems to have benefits both in terms of stability and performance on Acrobot and CartPole, however MountainCar does not exhibit the same improvements. Surprisingly, convolving on the constructed image in the reduced subspace performes as well as, if not better than, the inverse-transformed PCA models. In both cases, the PCA seems to improve learning over the original (D)DQN models by reducing the state space for relatively short learning times.

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

[2] Yasutaka Kishima and Kentarou Kurashige. Reduction of state space in reinforcement learning by sensor selection. Artificial Life and Robotics, 18(1-2):7–14, 2013.

[3] 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.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

[5] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. PyTorch, 2017.

[6] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In AAAI, volume 16, pages 2094–2100, 2016.