From Neural Fitted Q Learning to Deep Fitted Q and DQN

A brief discussion on neural fitted methods

Di Wu, 260562997

1 Introduction

After the deep-Q learning showed good performance in playing Atari game [1] and Alpha Go win the Go competition with Lee Sedol, deep reinforcement learning has become a hot research topic in the machine learning community. Deep reinforcement learning combines the metrics of both the conception power of deep learning and decision making power of reinforcement learning (RL), and shows great performance in serval applications. In deep reinforcement learning framework, deep learning can be used to approximate the value function, policy function or be used to predict the future reward or state representation.

In this course report, I am trying to provide a brief discussion on there highly related reinforcement learning methods including: neural fitted Q method, deep fitted Q [2], and Deep-Q learning [1]. The real world applications [3, 4] for fitted Q and deep fitted Q learning are also discussed.

In Section 2, the neural fitted Q framework is introduced. A neural fitted Q based home energy management application is also described. In Section 3, the deep fitted Q framework which use auto encoder for reinforcement learning is discussed. The application of deep fitted Q on autonomous driving is discussed. The Deep Q network is discussed in 4. Then the report is concluded in 5.

2 Neural Fitted Q Method

2.1 Neural Fitted Q

In [5], the authors propose a model free, neural network based reinforcement learning algorithm which use a multi-layer perceptron to represent the Q-value function. NFQ can be seen as another realization of fitted Q iteration [6]. The main idea is that for a markov decision process problem when the state is continuous the tabular reinforcement learning methods will not work, we will need to approximate the Q value function. This can be implemented with a neural network.

The main idea behind NFQ is that instead of updating the neural value function on-line, the updating can be performed off-line with an entire transition experience. There are mainly three steps for NFQ algorithm. The first step is exploration, we need to gather transition experience including tuples of $(s, a, r, s\prime)$. The second step is to train the neural network to approximate the Q function. The last step is to use the learned Q value function for control problems. With NFQ we can learn with a batch of transition experience which would be more data efficient.

2.2 Applications of Neural Fitted Q Method

In [4], the authors use NFQ algorithm to design control strategy for a home energy management system. For a smart home, the home energy management system need to minimize the operation cost while satisfying the customer's energy demand. In the paper, the energy supplies include solar energy and energy power buying from power grid, and the energy management system can store extra energy into a stand alone battery system when there is energy surplus. This can be modeled as a markov decision problem. The action is how much energy we sell for each time slot, and the state is the current energy demand, current electricity price, and current renewable energy generation. The reward is the the revenue for the energy sold to the power grid.

Algorithm 1 Neural Fitted Q

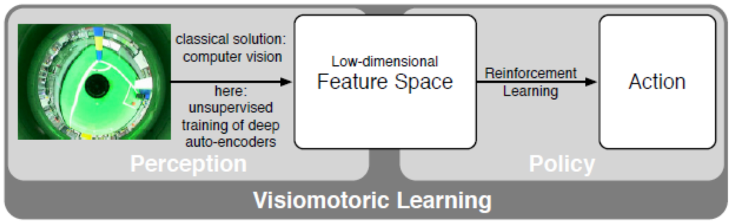
```
Input: Data set $\mathcal{S}$, kernel functions $\mathcal{K}$, number of iterations $T$
Require: Load F = $s(t), a(t), r(t), s(t+1)|t = 1, .., n$
Require: Define $\bar{Q}^0(s,a) = 0, \forall(s,a) \in F$, and $\bar{q}^h_{s,a} \in \bar{Q}^h(s,a)$
Require: Define H as the Horizon to be performed
Require: Define $TS^0$ as an initially empty training set
 1: h = 1;
 2: while h $\leq$ H do
 3:     for all $(s(t), a(t), r(t), s(t+1)) \in F$ do
 4:         $\bar{q}^h_{s,a} = r(t) + \gamma argmax_{a \forall As(t+1)} \bar{Q}^h(s(t+1), a)$
 5:         if $(s(t), a(t), .) \in TS^{h-1}$ then
 6:             $TS^{h-1} \leftarrow TS^{h-1} - \{(s(t), a(t), .)\}$
 7:         end if
 8:         $TS^{h-1} \leftarrow TS^{h-1} \bigcup \{(s(t), a(t), .)\}$
 9:     end for
10: end while
11: %Supervised learning%
12: Use supervised learning to train a function approximatior
13: $\bar{Q}^H(s,a)$ on the training set $TS^h$
14: h $\leftarrow h + 1$
15: % Obtain the policy%
16: for $\forall s \in S$ do
17:     $\pi(s) = argmax_{a \forall A_s} \bar{Q}^H(s,a), \forall s \in S$
18: end for
```

For this problem we usually would have some historical data and do not have a good simulator, NFQ algorithm would fit for this problem. Then we can first use the historical data to generate a batch of transition experience, and then use this transition experience to learn a good Q value function. Then finally in the execution phase we can use the learned Q function to perform energy management: in each time slot, we choose the action that can provide the largest reward.

3 Deep Fitted Q

3.1 Deep Fitted Q

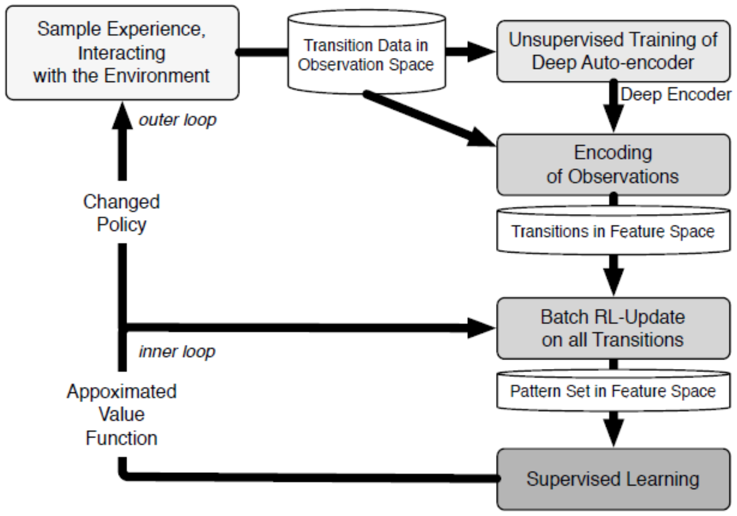
In [2], propose a deep fitted Q framework which is to use deep autoencoder neural networks for visual reinforcement learning tasks. For visual tasks the input will be raw images which are high dimension inputs. It is usually difficult for reinforcement learning framework to deal with raw image directly. Traditionally the process will include two steps as shown in Figure 1. The first step is to preprocess the images with some computer vision methods. In this step the high dimension inputs are extracted and condensed to low dimension inputs. The second step is to learn a reinforcement learning policy with low dimension inputs.



classical solution: computer vision
here: unsupervised training of deep auto-encoders
Low-dimensional Feature Space
Reinforcement Learning
Action
Perception
Policy
Visiomotoric Learning

Figure 1: Classic decomposition process for visual reinforcement learning tasks [2]

Deep fitted Q framework is shown in Figure 2. Deep fitted Q is built on neural fitted Q framework which can learn a control policy based on a set of experience and is data efficient. For Figure 2, the outer loop is to use the current optimal Q function to derive a control policy: eg. $\epsilon - greedy$ evaluation to collect experience. Then use current autoencoder to process the collected experience (observation) and generate experience features, and then we use neural fitted Q on these features to learn a current optimal Q function approximator. For each time, a new autoencoder need to be trained. Deep fitted Q framework propose an efficient solution to connect autoencoder for batch reinforcement learning.



Sample Experience, Interacting with the Environment
Transition Data in Observation Space
Unsupervised Training of Deep Auto-encoder
Deep Encoder
outer loop
Encoding of Observations
Changed Policy
Transitions in Feature Space
Batch RL-Update on all Transitions
inner loop
Appoximated Value Function
Pattern Set in Feature Space
Supervised Learning

Figure 2: Deep fitted Q framework [2]

3.2 Applications of Deep Fitted Q Method

In [3], the authors present a solid application for deep fitted Q learning. In this paper, the proposed system can learn a control policy with raw visual input and then derive a control policy for s lot car. For this task, the goal is to move the car in a given track as fast as possible. This control problem can be treated as the markov decision process. The state is current position and temporal information, reward is a small number for no crush, and great minus for crush (-1,000,000), action is the voltage that fed to the car control system. The system view is shown Figure 3. There mainly three steps for the control system. 1) Gather raw input data for an autonomously learned deep neural network, here raw data means only raw visual image inputs without any semantics instructions. 2) Autoencoder builds the state and temporal information with input data 3) Learn a control policy (Q function approximator) with neural fitted Q. Simulation results show that after 400 steps training, the learned policy could provided three time faster performance than baseline.

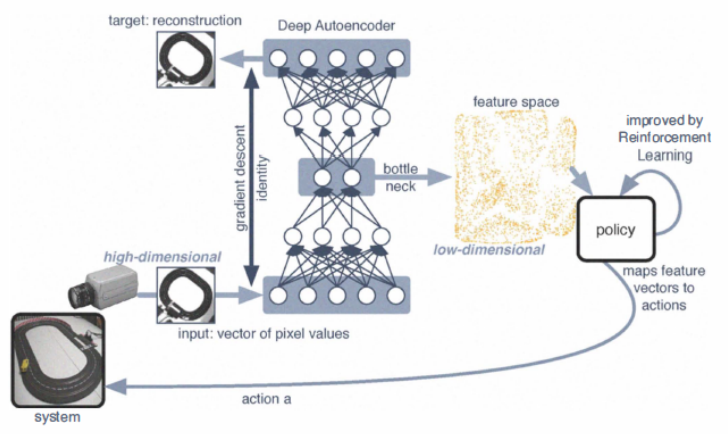
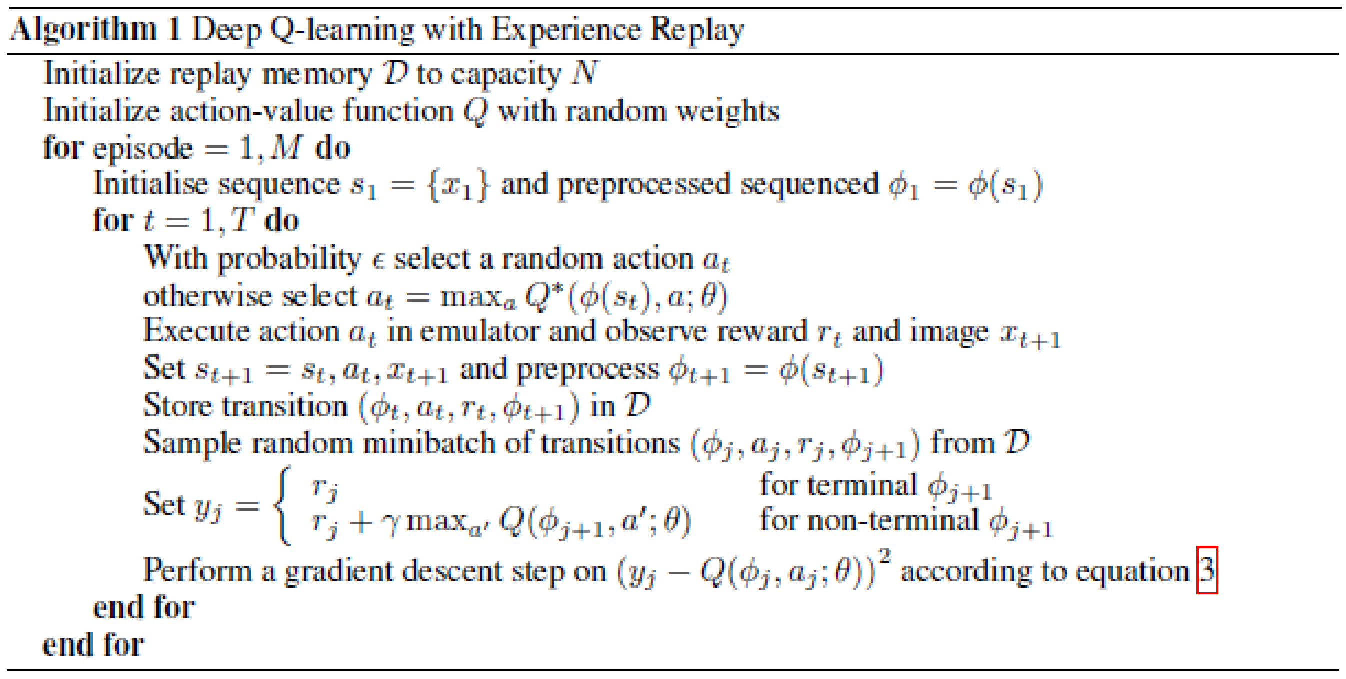


Fig. 3. Overview of the system.

Figure 3: System overview [3]

4 DQN Network

Deep learning have accomplished several breakthroughs in computer vision [7] and speech recognition [8, 9], and natural language processing tasks [10]. When trying to combine with reinforcement learning framework there are still some open challenges. First is that deep learning usually require a large amount of labelled data while reinforcement learning need to learn from a sequence of experience which is usually sparse, noisy, and delayed. Second is that deep learning usually assume the training data are independent while the data for reinforcement learning are highly correlated. The deep-Q learning proposed in [1], can overcome the mentioned problems. The pseudo code for deep-Q learning is shown in Figure. 4. It uses a convolutional neural network to learn control policies from raw video data. Simulations are presented for Atari 26000 and shows that the proposed algorithm outperform all previous approaches in six games and surpass one human expert on three games.



```
Algorithm 1 Deep Q-learning with Experience Replay
  Initialize replay memory $\mathcal{D}$ to capacity $N$
  Initialize action-value function $Q$ with random weights
  for episode $= 1, M$ do
      Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
      for $t = 1, T$ do
          With probability $\epsilon$ select a random action $a_t$
          otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
          Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
          Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
          Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
          Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
          Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
          Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
      end for
  end for
```

Figure 4: Pseudo code for deep-Q learning [1]

The most similar previous work for deep-Q leaning is neural fitted Q leaning [5] and deep fitted Q leaning [2]. For deep fitted Q learning it can also use raw visual data to learn a control policies [2, 3]. However there are several significant difference for deep-Q learning which makes it outperforms all pervious approaches.

- Deep-Q learning learn the policies in a online fashion and use stochastic decent to learn the control polices which will have a low constant cost per iteration. For batch reinforcement learning, it may have a computational cost per iteration.
- Deep-Q learning applies the reinforcement learning in a end-to-end mode while deep fitted Q learning would use deep learning and reinforcement learning in two stages which is somehow separate.
- Experience relay [11] is adopted in deep-Q learning which can break the dependency of training data and thus reduce the variance.

5 Conclusion

Deep reinforcement learning which combine deep learning into reinforcement learning framework is currently a hot research topic and develops very fast. In this paper, I briefly review the three most similar and highly related work: neural fitted Q learning, deep fitted Q learning and deep-Q learning.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[3] S. Lange, M. Riedmiller, and A. Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[4] H. Berlink and A. H. Costa. Batch Reinforcement Learning for Smart Home Energy Management. In *IJCAI*, pages 2561–2567, 2015.

[5] M. Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[6] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[7] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.

[8] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[9] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[10] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[11] L.-J. Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, Fujitsu Laboratories Ltd, 1993.