Applied Deep Learning: Final Project Report

Game Playing with DQfD and DQN

Instructors: Yun-Nung (Vivian) Chen and Hung-Yi Lee

Team: ***Praise The Sun***

深度強化學習（Deep reinforcement learning）是利用既有的強化學習演算法，結合近年來表現很好的深度學習，所形成的一類機器學習模型，目前在許多領域，例如電腦對局、機器人學、以及機器玩遊戲等，已有很好甚至超越人類的成果。不過，根據先前作業的經驗，傳統的深度強化學習模型在訓練期間頗為耗時，而且就算使用相同環境訓練相同模型，其訓練的成效有時也好壞不一。為了解決這樣的問題，（T. Hester et al., 2017）[1] 提出了 DQfD （Deep Q-Learning from Demonstrations）模型，並說明該模型可以僅僅藉由少量的遊戲演示資料來加速模型的訓練。本報告將以數種 Atari 遊戲作為訓練環境，比較 DQfD 以及 DQN（以及其變形）在這些遊戲中的學習狀態以及成果，並分析導致這些模型結果上差異的因素。

1 深度強化學習簡介

在強化學習中，一個模型裡存有一環境（environment）以及主體（agent），而主體的行為模式可視為一馬可夫決策過程（Markov desicion process，MDP）。MDP 可以以一個五元組 $(S, A, R(\cdot,\cdot), T(\cdot,\cdot,\cdot), \gamma)$ 表示。S 代表著所有狀態（state）的集合；A 為所有可能行動（action）的集合；$R(s,a)$ 為一獎勵函數（reward function）（即給定目前狀態 s 以及目前行動 a，其獎勵為多少）；$T\ s, a, s' = P(s'|s,a)$ 為一轉換函數（transition function），並服從某一機率分佈；γ 則為折減率（discount factor）。而主體在這個環境中，會根據某個策略函數 $\pi(s)$ 來行動。

在強化學習中，最常見的模型根據主體的性質主要分為基於策略（policy-based）以及基於價值（value-based）的模型：前者主要為直接尋找一個策略函數 π 使得其盡量接近最佳策略（即 $\pi \to \pi^*$）；而後者則是給定一個價值函數（$Q^\pi(s,a)$），該函數的目的即是估計出在目前狀態 s 下，若採取某行動 a 所能帶來的預期價值（expected value），而我們會希望該函數可以準確估計出各種狀況下的預期價值（即 $Q^\pi(s,a) \to Q^*(s,a)$），在這情況下，主體的最佳策略即為 $\pi^*(s) = \arg\max_{a \in A} Q^*(s,a)$。

Deep Q-Learning（DQN，直譯為「深度 Q 學習」）是一種常見的基於價值的深度強化學習模型。它的最佳價值函數可以以下面方程式（Bellman equation）表示：

$$Q^*(s,a) = \mathbb{E}\left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')\right]$$

在 DQN 裡，我們會用一個深度學習網路來代表 Q^π，而我們希望最後 Q^π 盡可能的接近 Q^*。在實務上，我們會用平均平方錯誤（mean squared error, MSE）作為訓練時的誤差函數，且為了穩定訓練，還會固定目標項之參數，如下所示：

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{R(s,a) + \gamma \max_{a'} Q(s',a',w^-)}_{\text{target, update slowly}} - \underbrace{Q(s,a,w)}_{\text{online, update quickly}}\right)^2\right]$$

其他常見的 DQN 模型還有：Double Q-Learning（DDQN）(H. van Hasselt et al., 2015) [3] 以及 Dueling Network （Z. Wang et al., 2015）[4] 等。前者認為，利用目標網路（target network）選取未來預期最大價值的行動，容易有過度估計上的誤差（upward bias）。因此，該模型修正誤差函數成如下：

$$\mathcal{L}(w) = \mathbb{E}\left[\left(R(s,a) + \gamma Q(s', \underbrace{\arg\max_{a'} Q(s',a',w)}_{\text{online network chooses optimal } a'}, w^-) - Q(s,a,w)\right)^2\right]$$

，而後者則將網路修正成：

$$Q(s,a,w) = \underbrace{V(s,w)}_{\textit{value}\text{, action-independent}} + \underbrace{A(s,a,w)}_{\textit{advantage}\text{, action-dependent}}$$

基本概念為：有些狀態天生較好（無論行動為何，預期價值一定比較高），而有些則否，因此預期價值可以視為是目前狀態天生的預期價值（獨立於行動），加上在該狀態下，不同動作下所能提昇/減少的價值。

2 DQfD：利用演示資料學習

DQfD 是一種混合了部份監督式學習（supervised learning）要素的強化學習演算法。相較於原始的 DQN，DQfD 在真正與環境互動訓練之前，會先使用一些事先收集好的演示資料（demonstration data）來進行預訓練（pre-training），之後才真正與環境互動進行強化。以真實的例子來比喻的話，這就有如一名運動員，先接受教練的訓練（專家演示）之後，才到比賽中（環境）累積經驗、強化自己的能力，而非直接在比賽中摸索。DQfD 除了多了一個預訓練的過程，在誤差的計算上，也和原始的 DQN 頗有不同。首先，為了確保模型有去學演示資料的動作，所以加了以下（監督式學習的）誤差函數：

$$\mathcal{L}_E(w) = \max_{a \in A}[Q(s,a,w) + l(a_E, a)] - Q(s, a_E), \text{ where } l(a_E, a) = \begin{cases} 0 & \text{, if } a = a_E \\ k & \text{, otherwise and } k \text{ is a positive number} \end{cases}$$

，這樣強制任何其他動作之預期價值會至少比 a_E 還要少 k，使得模型更傾向於去學演示資料的動作。除此之外，為了避免對於演示資料的過適（overfitting），對於網路的參數還會加一個 L2-regularization loss（$\mathcal{L}_{L2}(w)$）。最後，根據原始論文描述，為了讓模型符合原來的 Bellman equation，還會加上一個所謂的 n-step loss （$\mathcal{L}_n(w)$）。而總誤差為：

$$\mathcal{L}(w) = \underbrace{\mathcal{L}_{DQ}(w)}_{\text{original loss}} + \lambda_n \underbrace{\mathcal{L}_n(w)}_{\text{n-step loss}} + \lambda_E \underbrace{\mathcal{L}_E(w)}_{\text{supervised loss}} + \lambda_{L2} \underbrace{\mathcal{L}_{L2}(w)}_{\text{L2-regularization loss}}$$

額外設定 除此之外，在預訓練完之後，如一般的 DQN 一樣，模型將會把過去走過的紀錄存進一個暫存的記憶體中（即 replay buffer）。一般而言，該記憶體大小是有限的，而在 DQfD 中，雖然演示資料（demonstration）和探索資料（exploration data）都是存在裡面然後被定期採樣出來，但是演示資料永遠不會被刪除，而舊的探索紀錄則會。此外，論文中亦建議使用 prioritized replay buffer（T. Schaul et al., 2015）[2] 來讓演示資料有較高的優先權被採樣到。

因此，DQfD 相較於 DQN，總體上多了（一）演示資料（二）預訓練的過程（三）不同的誤差函數等設定。

DQN Network

type	activation	size	stride	output
input	-	-	-	$4 \times 84 \times 84$
conv	ReLU	8	4	$32 \times 20 \times 20$
conv	ReLU	4	2	$64 \times 9 \times 9$
conv	ReLU	3	1	$64 \times 7 \times 7$
flatten	-	-	-	3136
linear	ReLU	-	-	512
linear	-	-	-	(# of actions)
output	-	-	-	(# of actions)

Dueling DQN Network

type	activation	size	stride	output	remarks
input	-	-	-	$4 \times 84 \times 84$	
conv	ReLU	8	4	$32 \times 20 \times 20$	
conv	ReLU	4	2	$64 \times 9 \times 9$	
conv	ReLU	3	1	$64 \times 7 \times 7$	
flatten	-	-	-	3136	F
(value network)					
linear	ReLU	-	-	512	input: F
linear	-	-	-	1	
expand	-	-	-	(# of actions)	V
(advantage network)					
linear	ReLU	-	-	512	input: F
linear	-	-	-	(# of actions)	
zero-mean	-	-	-	(# of actions)	A
add	-	-	-	(# of actions)	V + A
output	-	-	-	(# of actions)	

表 1: DQN 及 Dueling Network 網路架構表，其中右圖中的 value network 以及 advantage network 是將 flatten 出來的結果作為輸入，而 value network 的輸出會複製成與 advantage network 維度相同的向量；advantage network 的輸出則會將其分佈從 (μ, σ^2) 標準化成 $(0, \sigma^2)$，以避免 value network 無效。

3 實驗設定

在這次報告中，我們將沿用 OpenAI 的 `gym` 作為訓練環境，並且選了其中 Atari 遊戲中的三款（Seaquest, Enduro, SpaceInvader）作為實驗環境。而模型將會接受一經過預處理過的，最終大小為 $4 \times 84 \times 84$ 的灰階圖片（其中的 4 則為目前狀態往前算起的四張灰階圖片）。本次實驗將比較下列模型/設定：

- 原始 DQN（vanilla DQN）
- Double DQN
- Dueling DQN
- DQfD，使用 n-step loss
- DQfD，不加 n-step loss

在進行訓練之前，我們首先蒐集了一些人類玩家遊玩的資料。主要作法為將 `gym` 裡的動作控制改為由人類來操控，並且在遊玩過程中，自動紀錄每步之狀態、動作、獎勵、以及下次的狀態等。為了讓人類較有時間反應，亦降低了遊戲進行的速度。人類玩家為本組中的其中三名組員，各自玩其中一款遊戲，並各自蒐集了 50000 步以上的資料。

表 1 為這次之網路架構圖，除了 Dueling DQN 以外皆使用左側的架構，而我們在 `pytorch 0.3.0` 上實作上述網路。optimizer 為 RMSProp，學習速率（learning rate）為 0.0001，折減率 γ 為 0.99，目標網路、訓練網路之更新頻率分別為每 1000 步一次以及每 4 步一次，replay buffer 大小為 10000，每次計算誤差時，從 replay buffer 取樣出 32 筆資料。

DQfD 中，首先會先預訓練 350000 次，取樣出演示資料的機率固定為 0.3，n-step loss 的 n 為 10，若使用 n-step loss 時 λ_n 為 1，λ_E 為 1，$l(a_E, a)$ 中的 k 為 0.8。

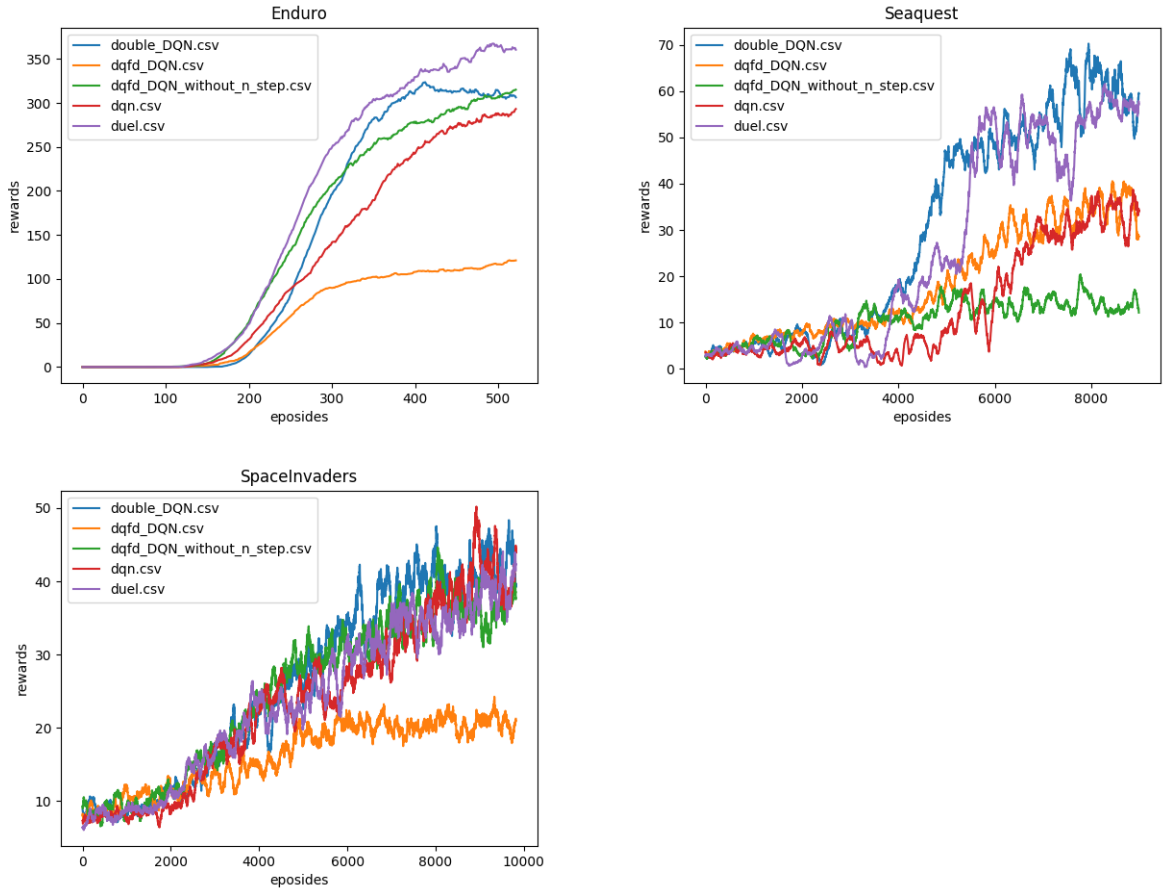

圖 1: 各模型之學習曲線。其中 Enduro 為一賽車遊戲，動作只有左、右、加速三種；而另外兩種比較偏向射擊遊戲，而 Seaquest 在複雜度上又比 SpaceInvaders 高。而 DQfD 之實作是以 DDQN 為基礎的，即其誤差函數中的 $\mathcal{L}_{DQ}(w)$ 是與 DDQN 的相同。

4 實驗結果與討論

圖 1 為各模型在不同遊戲上之學習曲線。很明顯的可以看見 DQfD 之表現未如預期。我們大概分析了導致無法超越 DDQN 等演算法的可能原因：

- 相較於原始論文的敘述，我們對記憶體中的資料做抽樣時，並未實作所謂的 Priortized Replay Buffer，即未讓演示資料有較高的優先權被抽到，導致了模型比較不會學到專家示範的資料。
- 在預訓練完成時，模型應該就要能夠玩出一點分數，但我們的卻沒有。主要原因是因為實作 DQfD 時，我們所設定的隨機探索排程與其他模型一樣，從完全隨機遞減到 5%。這樣會導致預訓練的結果可能會被隨機探索洗掉。
- 我們的模型在加了 n-step loss 時結果有些異常，可能有其他實作上的缺陷。

綜合以上分析，我們這次在實作上稱不上是成功。未來若要在繼續深入探討，可能可以從上述的問題開始下手。

參考文獻

[1] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[2] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[3] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[4] Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.