CS 234 H.W. #2

1.] The trajectory yielding max reward is given by:

t	(s_t, a_t, r_t, s_{t+1})	G_t ✓ total reward
1	(0, 2, 0, 2)	0
2	(2, 1, 2, 1)	2
3	(1, 2, 0, 2)	2
4	(2, 1, 2, 1)	4
5	(1, 0, 0.1, 0)	$\boxed{4.1}$

1.] Eqn (2) gives us

$$\vec{w} := \vec{w} + \alpha\left(r + \gamma \max_{a' \in A} \hat{q}(s', a', w) - \hat{q}(s, a, w)\right) \cdot \nabla_w \hat{q}(s, a, w)$$

As $\hat{q}(s, a, \vec{w}) = \vec{w}^T x(s, a)$ and $x(s, a)$ is simply a one-hot encoding of (state, action) pairs

$$\therefore \hat{q}(s, a, \vec{w}) = [w_{00}, w_{01}, w_{02}, \ldots, w_{sa}, \ldots] \cdot x(s, a)$$

$$= w_{sa} = Q(s, a)$$

(∴ equivalent to look-up table)

Further,

$$\nabla_{\vec{W}} \hat{q}(s,a,w) = \frac{\partial}{\partial \vec{W}} \left(\vec{W}^T x(s,a) \right)$$

$$= x(s,a)$$

$\therefore$ Eqn. (2) becomes

$$\vec{W} = \vec{W} + \alpha \left(r + \max_{a' \in A} Q(s',a') - Q(s,a) \right) \cdot x(s,a)$$

As $x(s,a)$ is one-hot vector, all terms but W_{sa} are 0:

$$\Rightarrow W_{sa} = W_{sa} + \alpha \left(r + \max_{a' \in A} Q(s',a') - Q(s,a) \right)$$

$$\Rightarrow Q(s,a) = Q(s,a) + \alpha \left(r + \max_{a' \in A} Q(s',a') - Q(s,a) \right)$$

$$= \text{Eqn. (1)}$$

2) So $\nabla_W \hat{q}(s,a,w) = \nabla_W (W^T x) = x$

$\therefore$ By eqn. (2),

$$W := W + \alpha \left(r + \max_{a' \in A} (W^T x) - W^T x \right) \cdot x$$

Scanned by CamScanner

$$= W + \alpha(r + w^T x - w^T x) \cdot x$$

$$= \boxed{W + \alpha r x}$$

$\therefore$ update rule is

$$\boxed{W := W + \alpha r x}$$

Scanned by CamScanner

5)

1.

The last 4 time steps are used as input to the network rather than just the most recent in order to capture velocity, acceleration, and other kinematical quantities of objects moving in these games.

2.

Experience replay benefits DQN in many ways including greater data efficiency (same experience may be used to perform many weight updates rather than just once as in online-Q learning) and breaking strong correlations between consecutive experiences to more effectively learn Q-function. A target network provides the benefit of stability in the training process by fixing the weights of this network and updating a different set of weights (actual training network) with respect to this fixed target rather than oscillating targets on every training step. This also helps in dealing with non-stationarity in the environment.

Scanned by CamScanner

3.)

The benefit of representing the Q function as a vector, with each entry corresponding to the q-value for a different action, is that fewer feedforward passes are required to compute the Q-values for all actions at once than compared to the alternative representation of $\hat{q}(s, a, w)$.

5.)

Comparing the plots, we see that more training time was required to attain quality performance but ultimately the deep-Q network yielded a better final score than linear approximation.

6.) • The first convolutional layer takes in a (80, 80, 4) input and convolves 32 8 x 8 filters with stride 4. As each filter introduces (8 · 8 · 4) weights and a bias, total # of parameters is 32 · (8 · 8 · 4 + 1)

= 8,224

Scanned by CamScanner

Similarly, the 2nd convolutional layer takes in output of layer 1 of dimensions (20, 20, 32) and convolves it w/ 64 4x4 filters of stride 2 resulting in

$$64 \cdot (4 * 4 * 32 + 1) = 32{,}832 \text{ parameter}$$

The 3rd convolutional layer takes in output of layer 2 w/ dimensions (10, 10, 64) and convolves it w/ 64 3x3 filters of stride 1, resulting in

$$64 \cdot (3 * 3 * 64 + 1) = 36{,}928 \text{ parameters}$$

Finally, the fully-connected layer takes in

$$64 \cdot (10 * 10 * 64) + 512 = 6401 * 512$$
$$= 3{,}277{,}312 \text{ parameters}$$

$\therefore$ Total # of parameters = sum of param in all layers = $\boxed{3{,}355{,}296 \text{ parameters}}$

Scanned by CamScanner

- There are generally more parameters to train in a DQN than a Q-network using linear approximation. If $\hat{q}_\theta(S, a, \vec{w}) = \vec{W}^T x(S, a)$, then number of parameters is simply the length of $\vec{w}$ (plus one for bias term of course).

7.

We note that the agent's score starts to level out relatively quickly (after ≈ first 2000 iterations) and oscillates around this limit.

Scanned by CamScanner

7.) 1.)

- Define

$$f(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}, \quad W = [W_1, W_2]$$

$$f(z) = \begin{cases} 1 & w_1 x_1 + w_2 x_2 + b > 0 \\ 0 & w_1 x_1 + w_2 x_2 + b \leq 0 \end{cases}$$

Proceeding on a case-by-case basis:

- $(x_1, x_2) = (0, 0), \Rightarrow b \leq 0$
- $(x_1, x_2) = (0, 1), \Rightarrow w_2 + b \leq 0$
- $(x_1, x_2) = (1, 0), \Rightarrow w_1 + b \leq 0$
- $(x_1, x_2) = (1, 1), \Rightarrow w_1 + w_2 + b > 0$

$\therefore$ Possible values satisfying above for AND operator are

$$\boxed{W = [1, 1], \quad b = -\frac{3}{2}}$$

- Proceeding similarly for OR;

(i) $(x_1, x_2) = (0, 0), \Rightarrow b \leq 0$

(ii) $(x_1, x_2) = (0, 1) \Rightarrow w_2 + b > 0$

(iii) $(x_1, x_2) = (1, 0) \Rightarrow w_1 + b > 0$

(iv) $(x_1, x_2) = (1, 1) \Rightarrow w_1 + w_2 + b > 0$

Scanned by CamScanner

$\therefore$ Satisfactory values are

$$\boxed{w = [1, 1],\ b = -\frac{1}{2}}$$

2.) XOR is impossible w/ given functional form. To prove this we again evaluate the possible inputs:

$(0,0)$: $b \leq 0$ [1]

$(1,1)$: $w_1 + w_2 + b \leq 0$ [2]

$(1,0)$: $w_1 + b > 0$ [3]

$(0,1)$: $w_2 + b > 0$ [4]

Note [2], [3], and [4] imply

$$b > -(w_1 + w_2 + b) > 0,$$

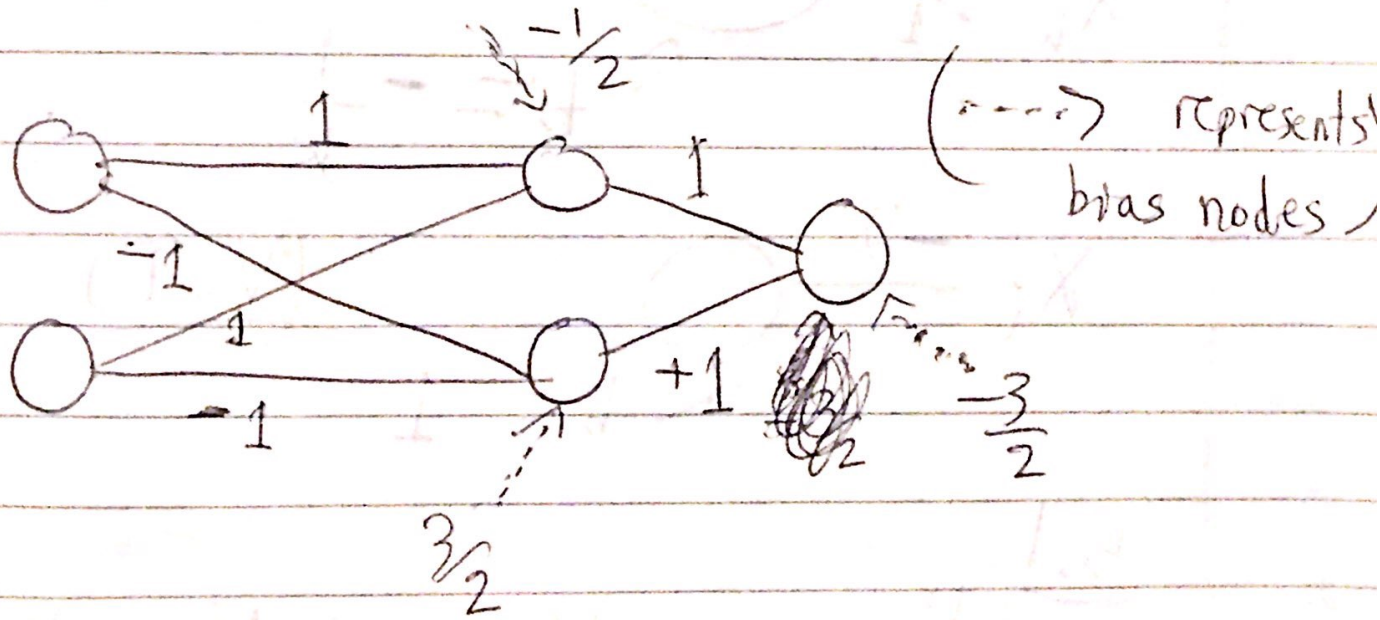
This contradicts [1] and hence no b, w_1, and w_2 can satisfy [1] – [4] to yield XOR

Scanned by CamScanner

3.] Any boolean function can be decomposed as ~~into~~ a nested operation of AND, OR, and NOT operations. Note that

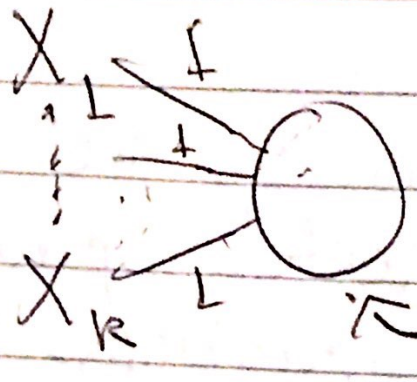
$$XOR(x_1, x_2) = NOT(AND(x_1, x_2))$$

Hence, we deduce the neural net



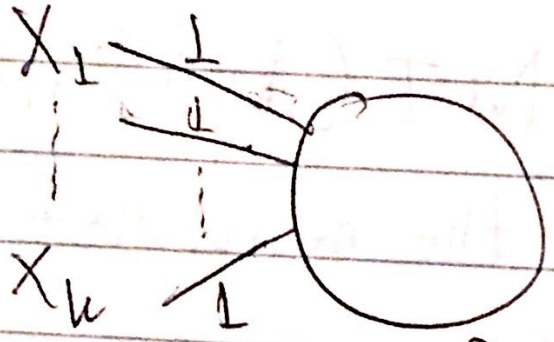
4.] As previously mentioned, any boolean function $f(x_1, \ldots, x_n)$ can be re-expressed into a decomposition of AND, OR, and NOT operations. Further, each of these can be constructed as neural nets as follows and stacked logically as in the XOR case for a single hidden layer:

Scanned by CamScanner



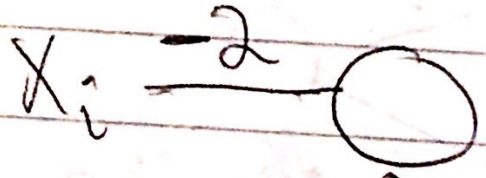
: (AND)

$b = -\left(\frac{2k-1}{2}\right)$



: (OR)

$b = -\frac{1}{k}$



: (NOT)

$b = 1$

5.)

Having multiple hidden layers allows the network to encode intermediate, higher-level features from the raw $\vec{x}$ and thus better representation than a single hidden layer, implying faster convergence and better stability.

Scanned by CamScanner