COMP9414 Tutorial

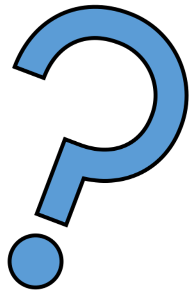
Week 10 (Final week)

News

- Assignment 2 marking will begin soon
  - Expect to see marks possibly next week

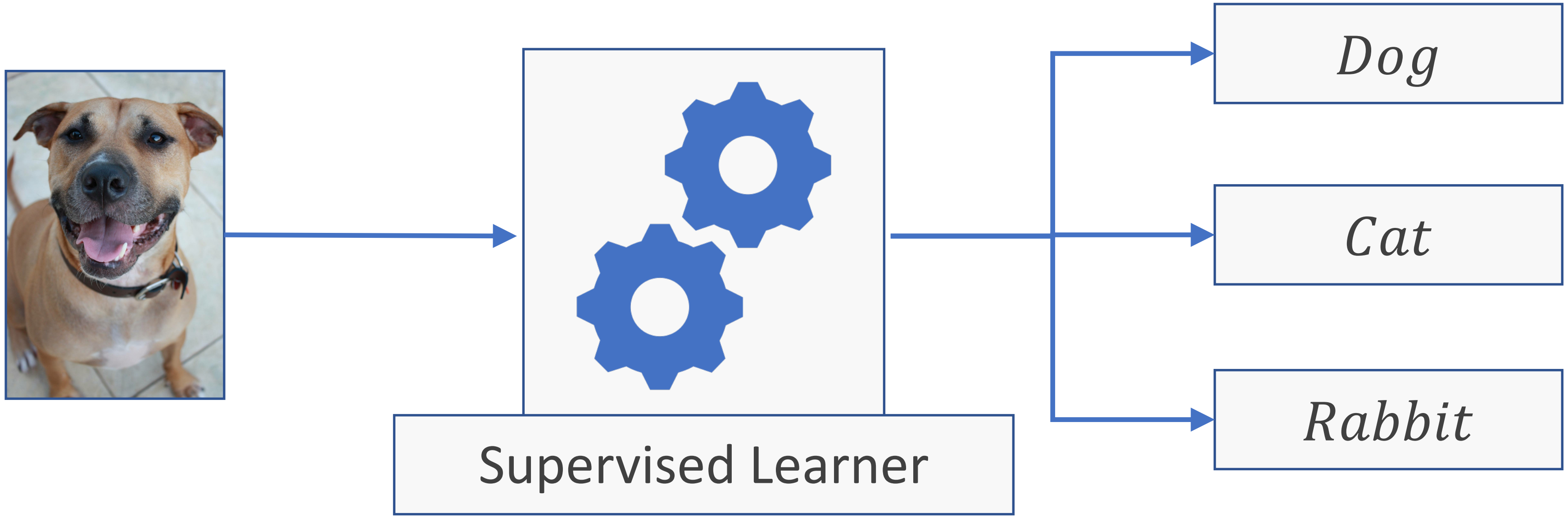
- Exam consultations on:
  - 2pm Tuesday (August 11th)
  - 2pm Thursday (August 13th)
  - 2pm Monday (August 17th)

- Exam is at 2 - 5pm Tuesday (August 18th)
  - 2 hours and 10 mins in length, must complete by 5pm
  - Make sure you do the practise exam to test your network



Supervised Learning

Train an algorithm to learn a mapping function between a set of input and output		
$(x_0, y_0), (x_1, y_1) \dots (x_n, y_n)$	$x_i \rightarrow f(x_i) \rightarrow y_i'$	$y_i' \cong y$



Perceptrons

$$\hat{y} = \begin{bmatrix} 1.0 \\ x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix} \cdot \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_k \end{bmatrix}$$

$$= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k$$

x = input matrix
w = weight matrix
y = actual result
$\hat{y}$ = predicted result

Generate a weight matrix that ensures: $\boxed{\hat{y} = y}$

$\boxed{w_0 \equiv b}$

Question 1a

Training Example	x_1	x_2	Class
a	0	1	-1
b	2	0	-1
c	1	1	1

Question 1a

Slope = m = 0 – 1 / 2 – 0 = -1/2

y = mx + b

1 = -1/2 * 0.5 + b

b = 1.25 = 5/4

x_2 = -0.5x_1 + 1.25

0 = 0.5x_1 + x_2 - 1.25

0 = 2x_1 + 4x_2 - 5

Question 1a

Training Example	x_1	x_2	Class
a	0	1	-1
b	2	0	-1
c	1	1	1

$0 = 2x_1 + 4x_2 - 5$	
w_0	-5
w_1	2
w_2	4

Question 1b Perceptron Learning Algorithm

1	Calculate s (prediction)
2	Execute transfer function
3	Compare prediction with actual class
4	Change weights if necessary

Example	x_1	x_2	Class
a	0	1	-1
b	2	0	-1
c	1	1	1

w_0	w_1	w_2
-0.5	0	1

Question 1b Perceptron Learning Algorithm

1	Calculate s (prediction)

$s = w_0 + w_1 x_1 + w_2 x_2$		
w_0		-0.5
$w_1 x_1$	$0 * 0$	0
$w_2 x_2$	$1 * 1$	1

$w_0 + w_1 x_1 + w_2 x_2$	0.5

w_0	w_1	w_2
-0.5	0	1

Example	x_1	x_2	Class
a	0	1	-1

Question 1b Perceptron Learning Algorithm

2	Execute transfer function $g(s)$

$g(s)$ {

$if\ s \geq 0$

$then\ 1$

$if\ s < 0$

$then\ 0$

}

w_0	w_1	w_2
-0.5	0	1

Example	x_1	x_2	Class
a	0	1	-1

$g(0.5)$	1

Question 1b Perceptron Learning Algorithm

3	Compare prediction with actual class

$if\ g(s) > g(actual\_class)$		
	$Subtraction$	-
$if\ g(s) < g(actual\_class)$		
	$Addition$	+
	$if\ 1 > 0$	-

w_0	w_1	w_2
-0.5	0	1

Example	x_1	x_2	Class
a	0	1	-1

Question 1b Perceptron Learning Algorithm

4	Change weights if necessary

Addition

$$w_0 = w_0 + learning\_rate$$

$$w_k = w_k + (learning\_rate \ * x_k)$$

Subtraction

$$w_0 = w_0 - learning\_rate$$

$$w_k = w_k - (learning\_rate \ * x_k)$$

w_0	w_1	w_2
-0.5	0	1

Example	x_1	x_2	Class
a	0	1	-1

$learning\_rate$	1

Question 1b - Learning Rate

Control the degree that the weights can be changed

Larger learning rate results in big changes

Converges to the optimal solution faster, but risks moving past or losing it

$s = w_0 + w_0 x_1 = 1 + 5 * 10 = 51$	$\hat{y} = 51, y = 55$	$\hat{y} \neq y$	$Addition$

$w_0 = 1$
$w_1 = 5$
$x_1 = 10$
$Learning\_rate = 4$

$w_0 = w_0 + 4$	5	Completely missed the predicted value.
$w_1 = w_1 + (4 * 10)$	45	
$s = 5 + 45 * 10 = 5 + 450 = 455$		

Question 1b - Learning Rate

Smaller learning rate results in shorter, more precise steps

Converges to the optimal solution much slower, but can get closer since the steps are miniscule

$s = w_0 + w_0 x_1 = 1 + 5 * 10 = 51$	$\hat{y} = 51, y = 55$	$\hat{y} \neq y$	$Addition$

$w_0 = 1$
$w_1 = 5$
$x_1 = 10$
$Learning\_rate = 0.005$

$w_0 = w_0 + 0.005$	1.005	Step is too small, will take too long to converge.
$w_1 = w_1 + (0.005 * 10)$	5.05	

$$s = 1.005 + 5.05 * 10 = 1.005 + 50.5 = 51.505$$

Question 1b Perceptron Learning Algorithm

4	Change weights if necessary

Example	x_1	x_2	Class
a	0	1	-1

$Subtraction$

$w_0 = w_0 - learning\_rate$	$w_0 = -0.5 - 1$	-1.5
$w_k = w_k - (learning\_rate \; * x_k)$	$w_1 = 0 - (1 * 0)$	0
	$w_2 = 1 - (1 * 1)$	0

w_0	w_1	w_2
-0.5	0	1

$\rightarrow$

w_0	w_1	w_2
-1.5	0	0

Question 1b

Training Example	x_1	x_2	Class	$g(Class)$
a	0	1	-1	0
b	2	0	-1	0
c	1	1	1	1

w_0	w_1	w_2
-0.5	0	1

Iteration 1	a	Prediction	+ 0.5	$g(0.5) = 1$	Subtraction
$w_0 = -0.5 - 1.0$					-1.5
$w_1 = 0.0 - (1.0 \times 0.0)$					0.0
$w_2 = 1.0 - (1.0 \times 1.0)$					0.0

Question 1b

Training Example	x_1	x_2	Class	$g(Class)$
a	0	1	-1	0
b	2	0	-1	0
c	1	1	1	1

w_0	w_1	w_2
-1.5	0	0

Iteration 2	b	Prediction	- 0.5	$g(-0.5) = 0$	Subtraction
Do nothing since the predicted result matches the positivity of the actual result					

Question 1b

Training Example	x_1	x_2	Class	$g(Class)$
a	0	1	-1	0
b	2	0	-1	0
c	1	1	1	1

w_0	w_1	w_2
-1.5	0	0

Iteration 3	c	Prediction	- 1.5	$g(-1.5) \ = 0$	Subtraction
$w_0 = -1.5 + 1.0$					-0.5
$w_1 = 0.0 + (1.0 \times 1.0)$					1.0
$w_2 = 0.0 + (1.0 \times 1.0)$					1.0

Question 1b

Training Example	x_1	x_2	Class	$g(Class)$
a	0	1	-1	0
b	2	0	-1	0
c	1	1	1	1

w_0	w_1	w_2
-3.5	0	1

Iteration 21	c	Prediction	- 1.5	$g(-1.5) \; = 0$	Subtraction
$w_0 = -3.5 + (1.0 \times 1.0)$					-2.5
$w_1 = 0.0 + (1.0 \times 1.0)$					1.0
$w_2 = 1.0 + (1.0 \times 1.0)$					2.0

Question 2a

Perceptron to compute the OR function of m inputs

Set the bias weight to -1⁄2, all other weights to 1

m inputs where at least one must be true for the overall function to be true

a V b V c V d V e … V z

Question 2b

Perceptron to compute the AND function of m inputs

Set the bias weight to 1⁄2 - n, all other weights to 1

m inputs where all must be true for the overall function to be true

$a \wedge b \wedge c \wedge d \wedge e \ldots \wedge z$

Question 2c

2-Layer Neural Network to compute any (given) logical expression, assuming it is written in Conjunctive Normal Form.

Conjunctive normal form is a series of disjunctive predicates connected by conjunctions

$(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$

Question 2c

$(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$

Weights should be +1 for normal predicates

Weights should be -1 for negations

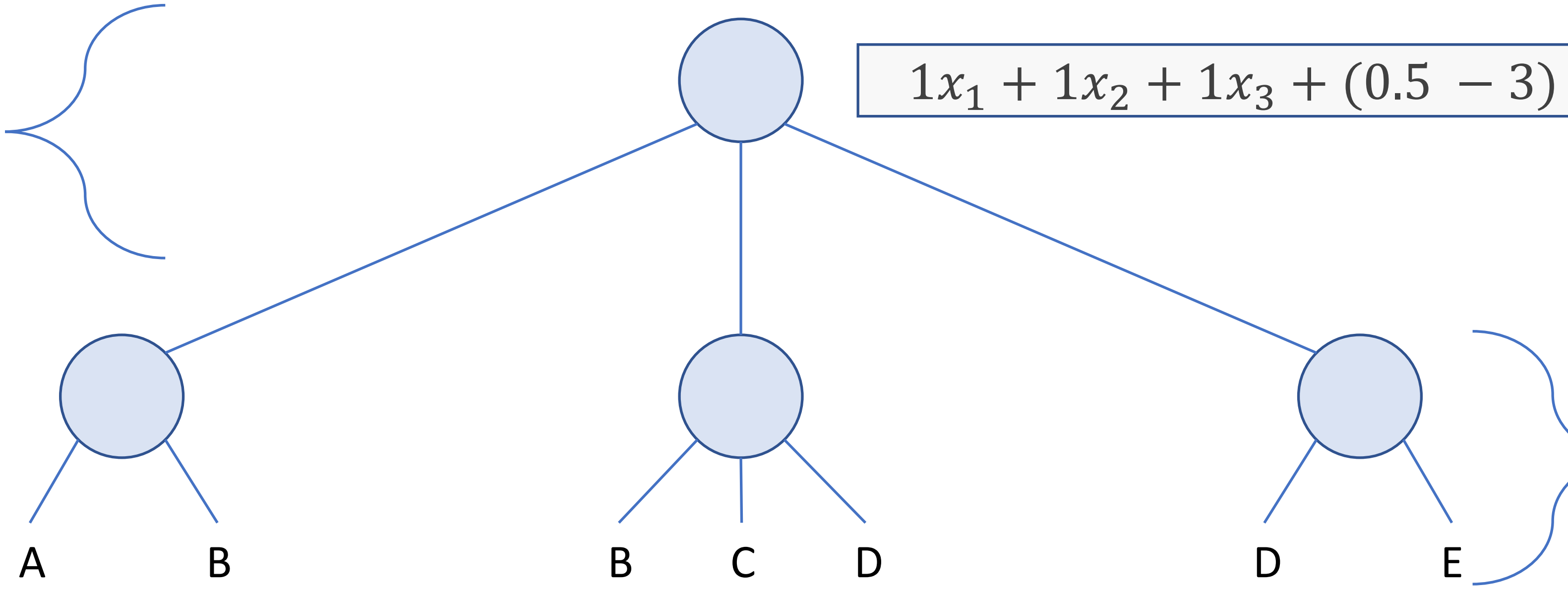
Basically a combination of the previous two questions

Question 2c

(A ∨ B) ∧ (¬B ∨ C ∨ ¬D) ∧ (D ∨ ¬E)

Conjunctive bias

$1x_1 + 1x_2 + 1x_3 + (0.5 \ - 3)$



Disjunctive bias

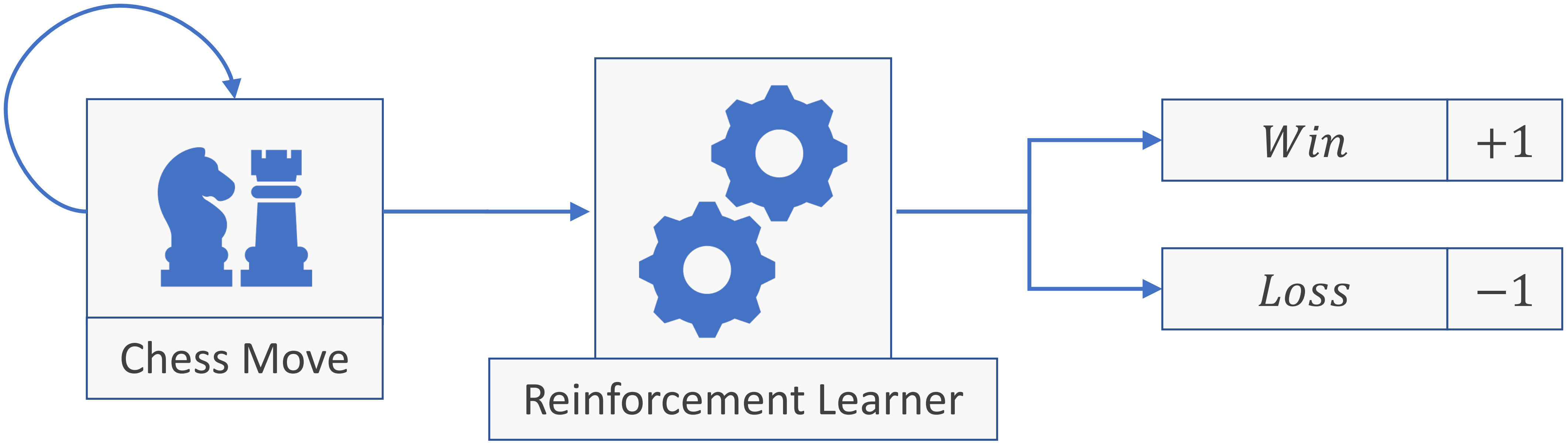
$1x_1 + 1x_2 + (0 \ - 0.5)$

$-1x_1 + 1x_2 - 1x_3 + \ (2 \ - 0.5)$

$1x_1 - 1x_2 + \ (1 \ - 0.5)$

Reinforcement Learning

Train an algorithm by providing it rewards when it makes the correct decision



Question 3i

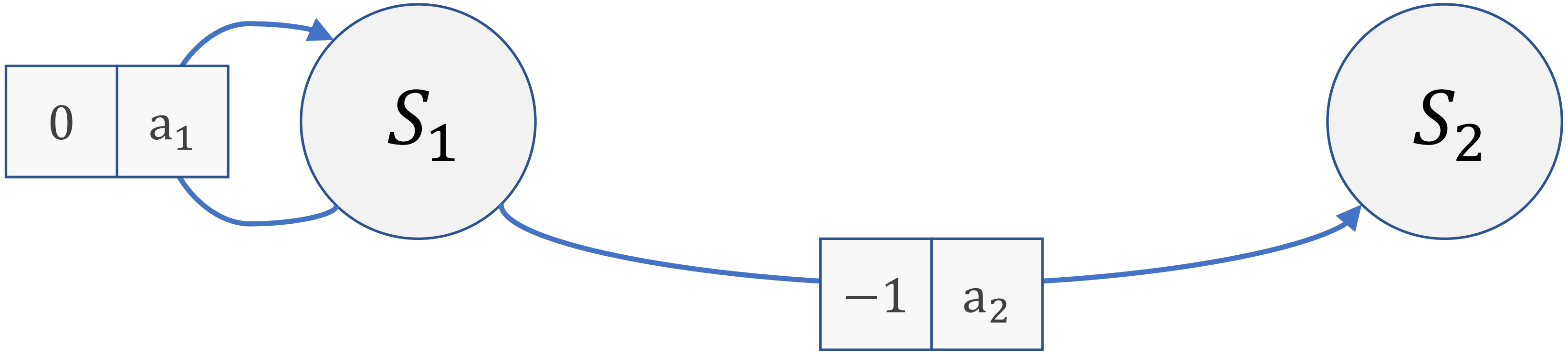
$S = \{S_1, S_2\}$	$A = \{a1, a2\}$

$\delta(S_1, a_1) = S_1$	$r(S_1, a_1) = 0$

Question 3i

$S = \{S_1, S_2\}$	$A = \{a1, a2\}$

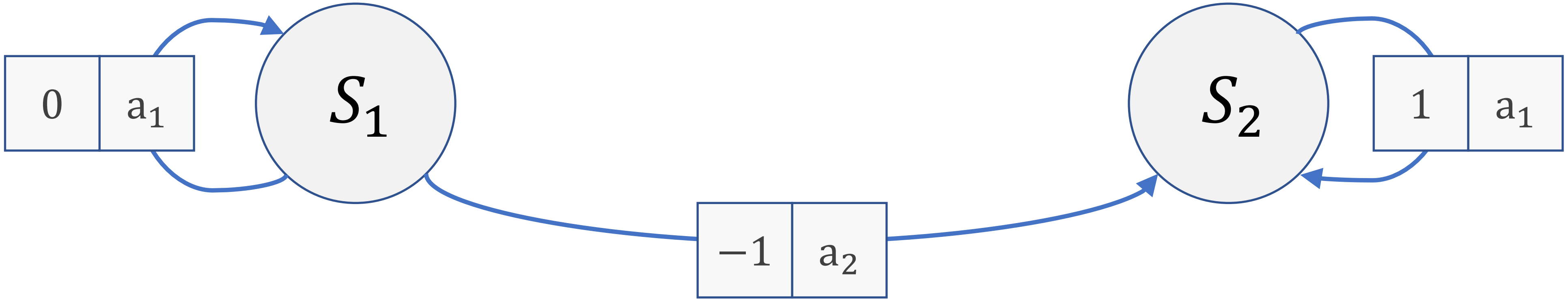
$\delta(S_1, a_2) = S_2$	$r(S_1, a_2) = -1$



Question 3i

$S = \{S_1, S_2\}$	$A = \{a1, a2\}$

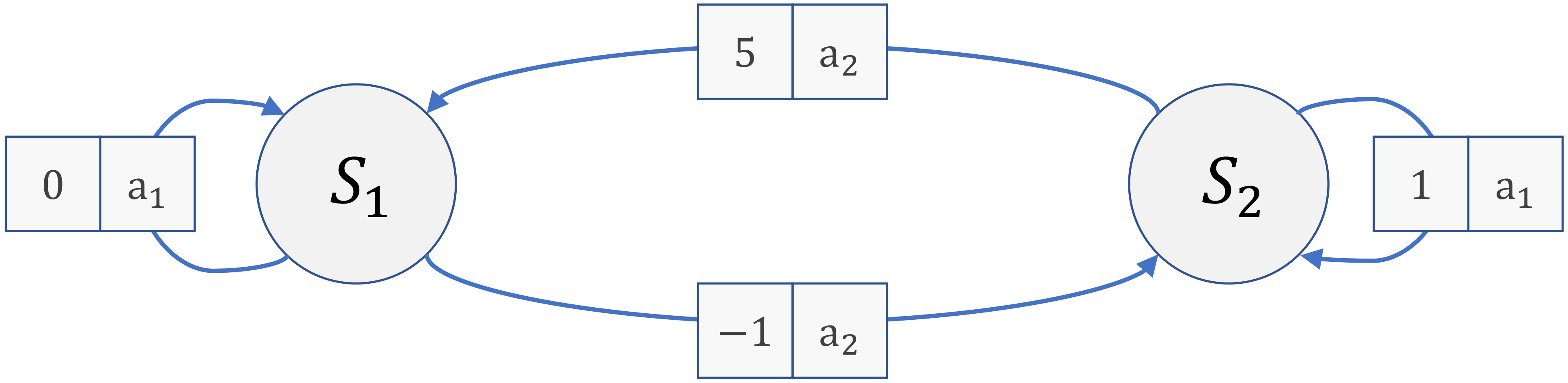
$\delta(S_2, a_1) = S_2$	$r(S_2, a_1) = 1$



Question 3i

$S = \{S_1, S_2\}$ | $A = \{a1, a2\}$

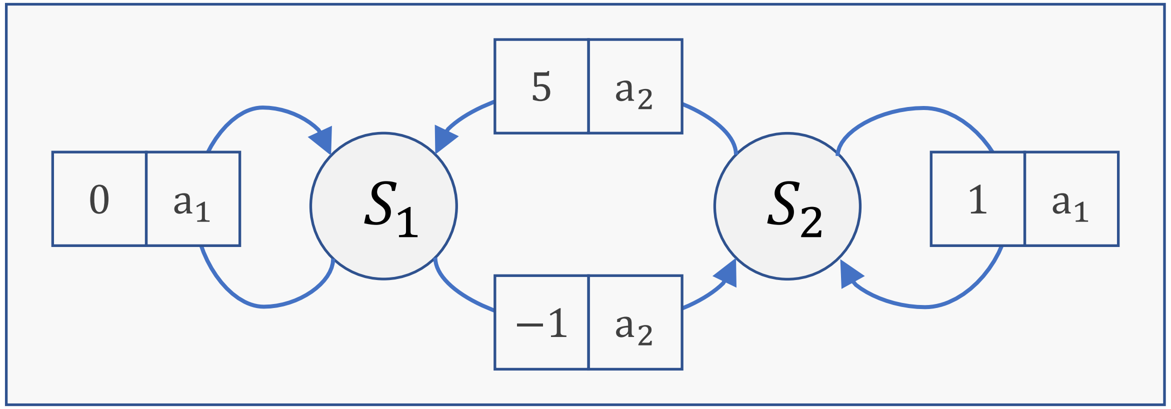
$\delta(S_2, a_2) = S_1$ | $r(S_2, a_2) = 5$



Question 3iib - Calculating the Optimal Policy

Bellman equation

$V^*(s) = r(s,a) + \gamma V^* \delta(s,a)$



$\pi: S \rightarrow A$

Optimal Policy	Maximises the cumulative reward

1	Calculate bellman equation for each state and action pair
2	Select the pair that achieves the highest combined result

This becomes the optimal policy

Question 3iib

$\gamma = 0.9$ | $V^*(s) = r(s,a) + \gamma V^* \delta(s,a)$

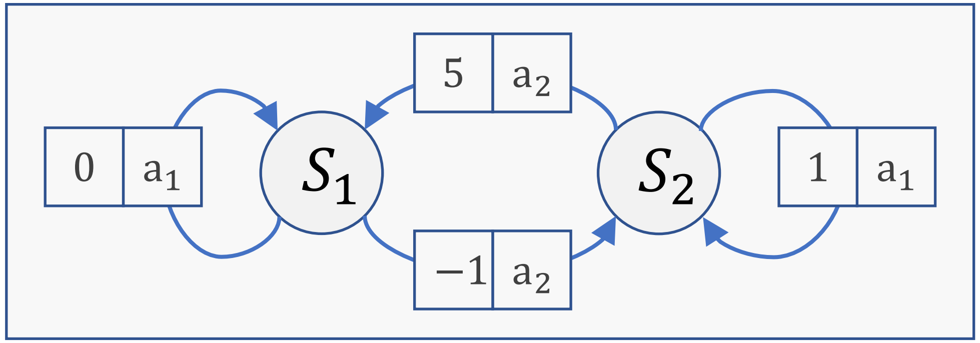
$\delta^*(S_1) = S_1$

$V^*(S_1) = 0 + 0.9V^*(S_1)$	$0.1V^*(S_1) = 0$	$10 * 0 = 0$

$\delta^*(S_2) = S_2$

$V^*(S_2) = 1 + 0.9V^*(S_2)$	$0.1V^*(S_2) = 1$	$10 * 1 = 10$

$V^*(S_1) = 0$
$V^*(S_2) = 10$



Question 3iib

$\gamma = 0.9$ | $V^*(s) = r(s,a) + \gamma V^* \delta(s,a)$

$\delta^*(S_2) = S_2$

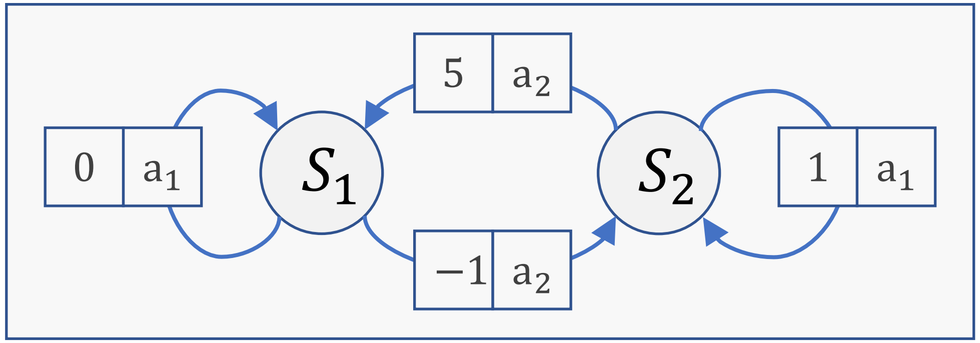
$V^*(S_2) = 1 + 0.9V^*(S_2)$ | $0.1V^*(S_2) = 1$ | $10 * 1 = 10$

$\delta^*(S_1) = S_2$

$V^*(S_1) = -1 + 0.9V^*(S_2)$ | $V^*(S_1) = -1 + (0.9 * 10)$ | $= 8$

$V^*(S_1) = 10$

$V^*(S_2) = 8$



Question 3iib

$\gamma = 0.9$ | $V^*(s) = r(s,a) + \gamma V^* \delta(s,a)$

$\delta^*(S_1) = S_2$

$V^*(S_1) = -1 + 0.9V^*(S_2)$

$= -1 + 0.9(5 + 0.9V^*(S_1))$ $\quad \cong r(S_1, a_2) + \gamma(\gamma V^* \delta(S_1, a_2))$

$= -1 + 4.5 + 0.9^2 V^*(S_1)$

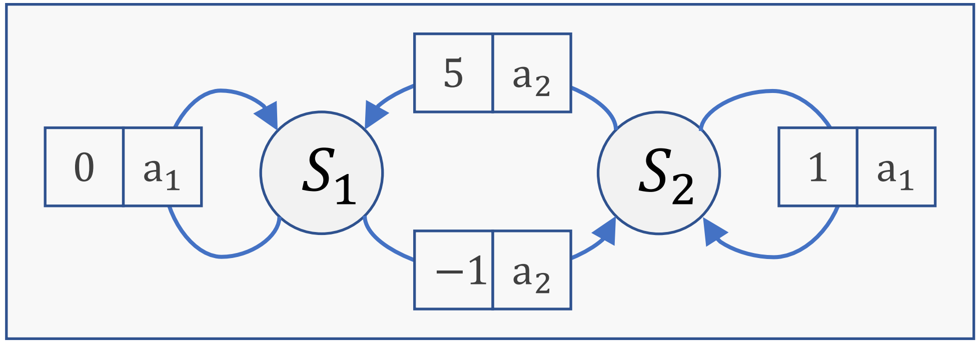
$= 3.5 + 0.9^2 V^*(S_1)$

$3.5 + 0.81V^*(S_1)$

$0.19V^*(S_1) = 3.5$

$V^*(S_1) = 3.5/0.19 = 18.42$

$\delta^*(S_2) = S_1$ | $V^*(S_2) = 5 + 0.9V^*(S_1)$



Question 3iib

$\gamma = 0.9$

$V^*(s) = r(s,a) + \gamma V^* \delta(s,a)$

$\delta^*(S_2) = S_1$

$V^*(S_2) = 5 + 0.9V^*(S_1)$

$= 5 + 0.9 * 18.42$

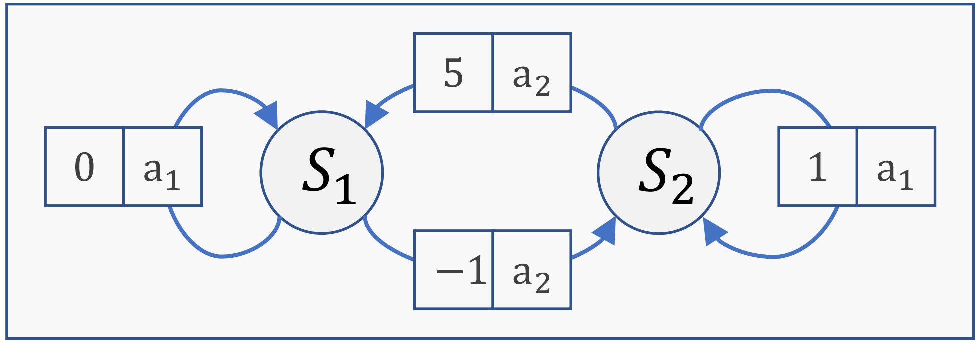
$= 5 + 16.58$

$V^*(S_2) = 21.58$

$V^*(S_1) = 18.42$

$V^*(S_1) = 18.42$

$V^*(S_2) = 21.58$



Question 3iic

$r(s,a) + \gamma V^* \delta(s,a)$

$Q(S_1, a_1)$	$0 + \gamma V^*(S_1)$	16.58
$Q(S_1, a_2)$	$V^*(S_1)$	18.42

$Q(S_2, a_1)$	$1 + \gamma V^*(S_1)$	20.42
$Q(S_2, a_2)$	$V^*(S_1)$	21.58

Note	Don’t need Gamma symbol since we already used it in the a_2 calculations for question 3iib.

Question 3iii

Q	a_1	a_2
S_1	16.58	18.42
S_2	20.42	21.58

Use Q-Learning to learn this optimal policy to progress between states in such a way to achieve the greatest value.

Question 3iv – Q-Learning

Values initially set to 0 or some fixed amount

Q	a_1	a_2
S_1	0	0
S_2	0	0

Quality learning learns a policy that tell an agent what action it should take given the current state

Aims to maximise the received reward over all successive steps

Question 3iv – Q-Learning

Loop until converge

Select an action a_t

Update the $Q(S_t, a_t)$ value corresponding to it

$Q(S_t, a_t) = r(S_t, a_t) + \ \gamma * (max_{0\ldots k} \ (\ Q(S_{t+1}, a_k))$

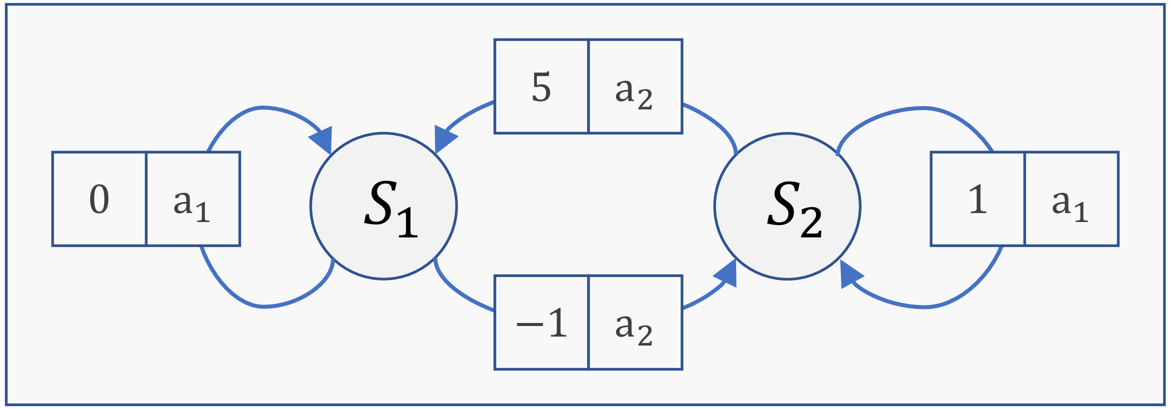
Where k is every possible actions available from the new state

Switch to the new state S_{t+1}

Q	a_1	a_2
S_1	0	0
S_2	0	0

Question 3iv – Q-Learning

Actions are selected using some degree of probability to encourage exploration	
Exploration	Select an action with a worse reward to determine if it leads to a potentially better one
Exploitation	Select an action with an already known reward to maximise the current cumulative reward

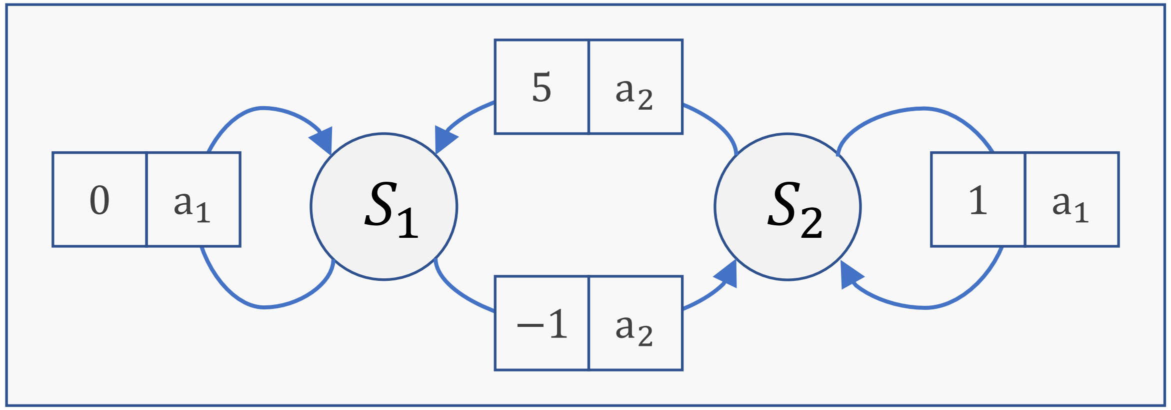


S_1
Exploration would select a_2 as the action
Exploitation would select a_1 as the action

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_1	a_1	$r(S_1, a_1) + \gamma * max(Q(S_1, a_1), Q(S_1, a_2))$	
		$0 + 0.9 * 0$	$= 0$

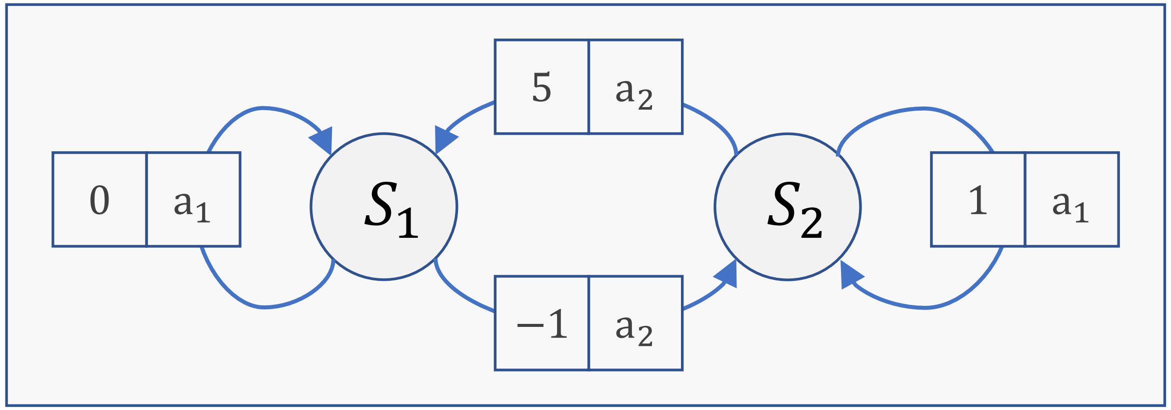


Q	a_1	a_2
S_1	0	0
S_2	0	0

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_1	a_2	$\mathrm{r}(S_1, a_2) + \gamma * max(Q(S_2, a_1), Q(S_2, a_2))$	
		$-1 + 0.9 * 0$	$= -1$

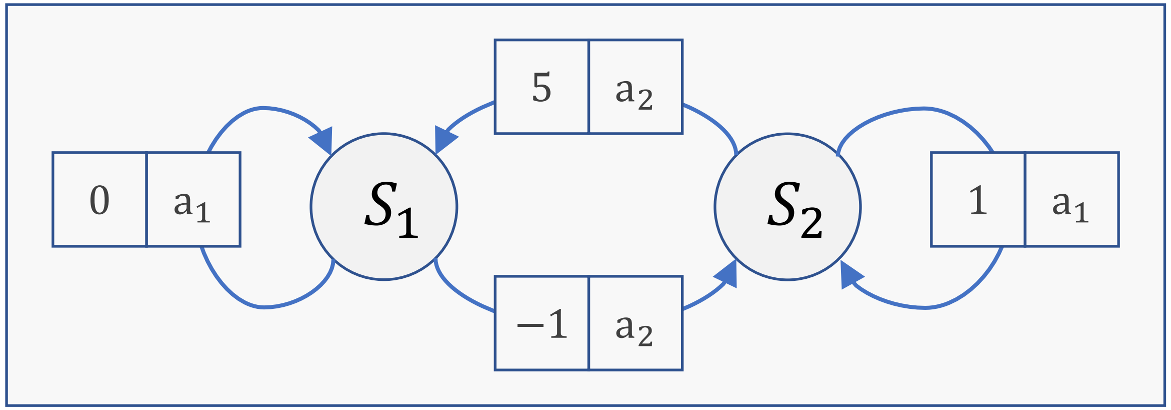


Q	a_1	a_2
S_1	0	-1
S_2	0	0

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_2	a_1	$r(S_2, a_1) + \gamma * max(Q(S_2, a_1), Q(S_2, a_2))$	
		$1 + 0.9 * 0$	$= 1$

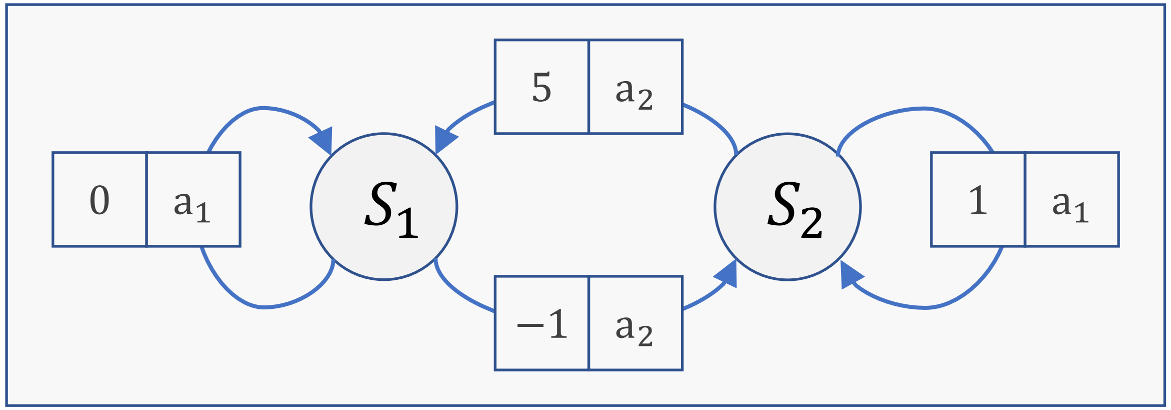


Q	a_1	a_2
S_1	0	-1
S_2	1	0

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_2	a_2	$\mathrm{r}(S_2, a_2) + \gamma * max(Q(S_1, a_1), Q(S_1, a_2))$	
		$5 + 0.9 * 0$	$= 5$

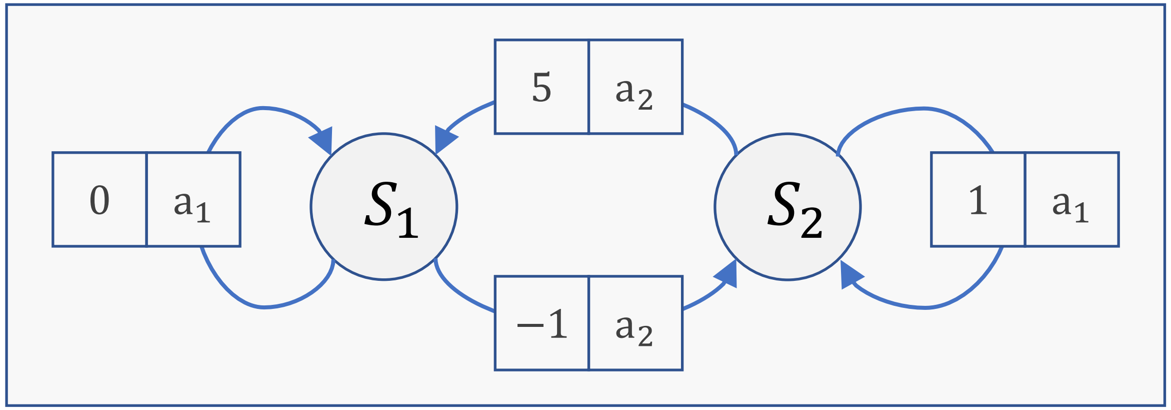


Q	a_1	a_2
S_1	0	-1
S_2	1	5

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_1	a_1	$\mathrm{r}(S_1, a_1) + \gamma * max(Q(S_1, a_1), Q(S_1, a_2))$	
		$0 + 0.9 * 0$	$= 0$

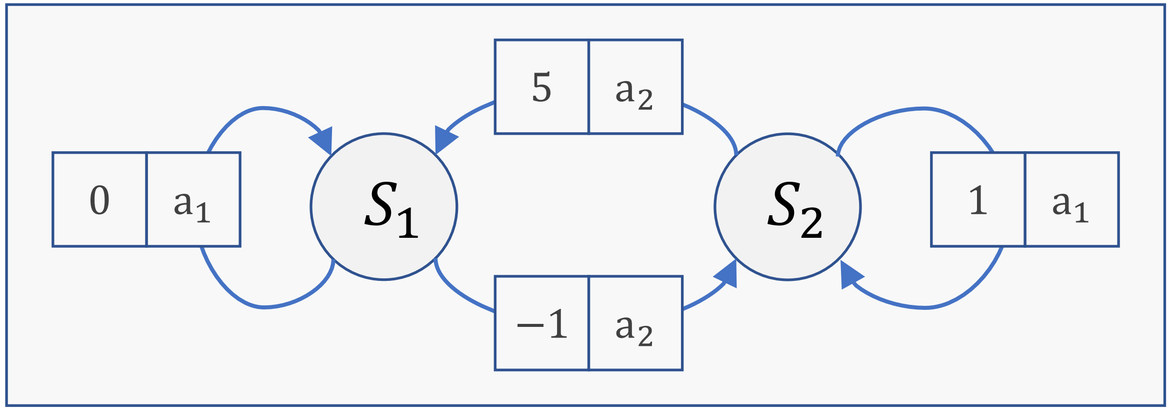


Q	a_1	a_2
S_1	0	-1
S_2	1	5

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_1	a_2	$r(S_1, a_2) + \gamma * max(Q(S_2, a_1), Q(S_2, a_2))$	
		$-1 + 0.9 * 5$	$= 3.5$

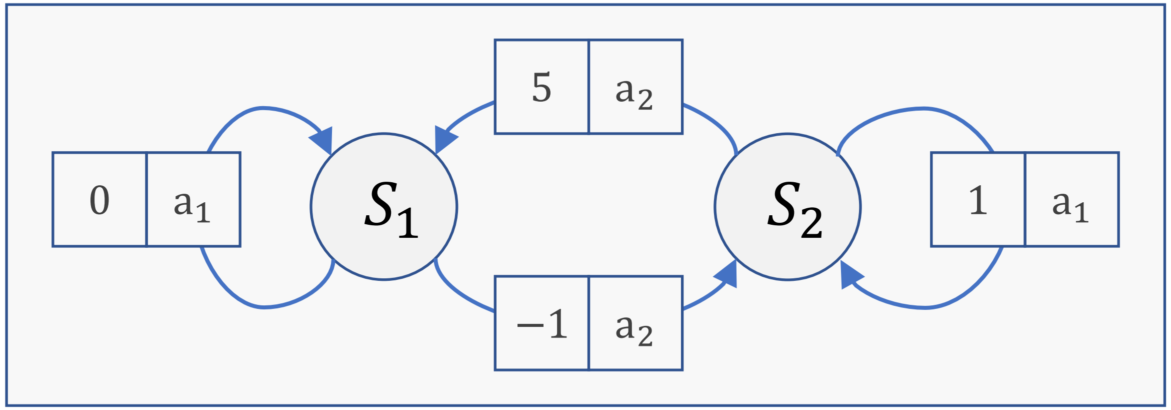


Q	a_1	a_2
S_1	0	3.5
S_2	1	5

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_2	a_1	$r(S_1, a_2) + \gamma * max(Q(S_2, a_1), Q(S_2, a_2))$	
		$1 + 0.9 * 5$	$= 5.5$

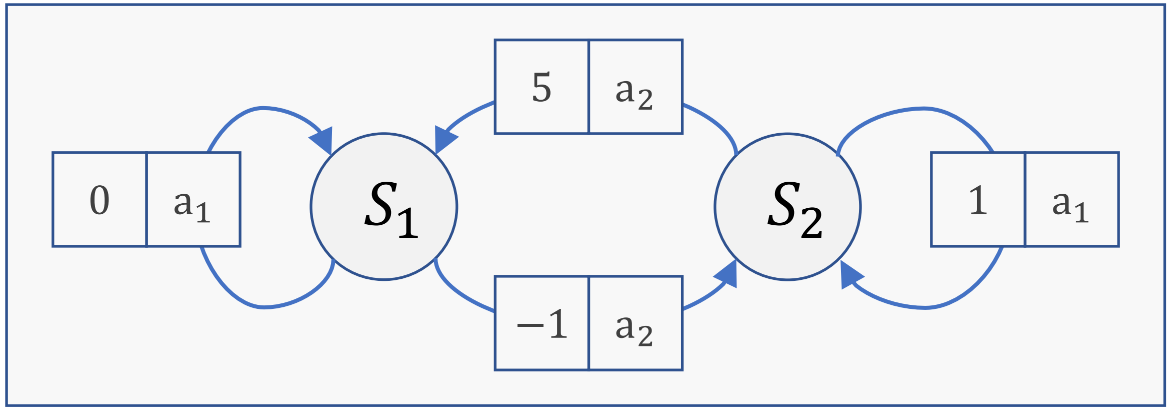


Q	a_1	a_2
S_1	0	3.5
S_2	5.5	5

Question 3iv – Q-Learning

$\gamma = 0.9$

State	Action	Updated Q Value	
S_2	a_2	$r(S_2, a_2) + \gamma * max(Q(S_1, a_1), Q(S_1, a_2))$	
		$5 + 0.9 * 3.5$	$= 8.15$



Q	a_1	a_2
S_1	0	3.5
S_2	5.5	5

Notes

- Q-Learning walkthrough (ignore the code)
  - http://mnemstudio.org/path-finding-q-learning-tutorial.htm

Good luck with the exam everyone!	Was nice to virtually teach you ~