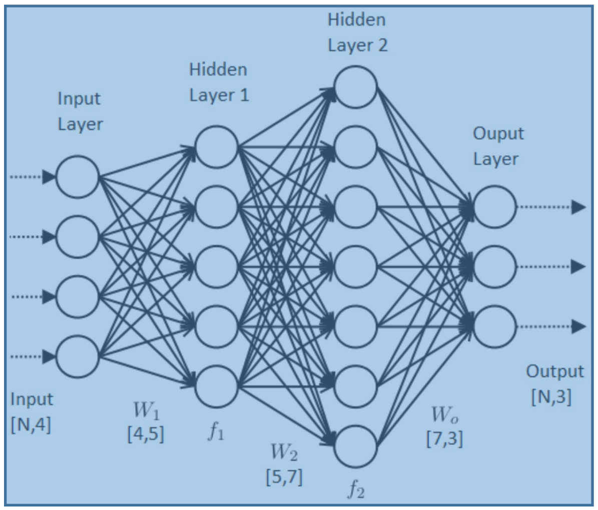
CSCE 636 Neural Networks (Deep Learning)

Lecture 13: Deep Reinforcement Learning (continued)

Anxiao (Andrew) Jiang

Policy-based Approach

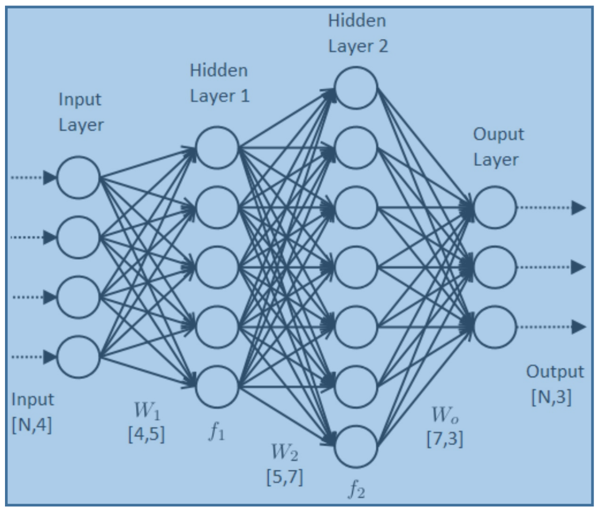
$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n, \theta)$$



Input
Layer
Hidden
Layer 1
Hidden
Layer 2
Ouput
Layer
Input
[N,4]
W_1
[4,5]
f_1
W_2
[5,7]
f_2
W_o
[7,3]
Output
[N,3]

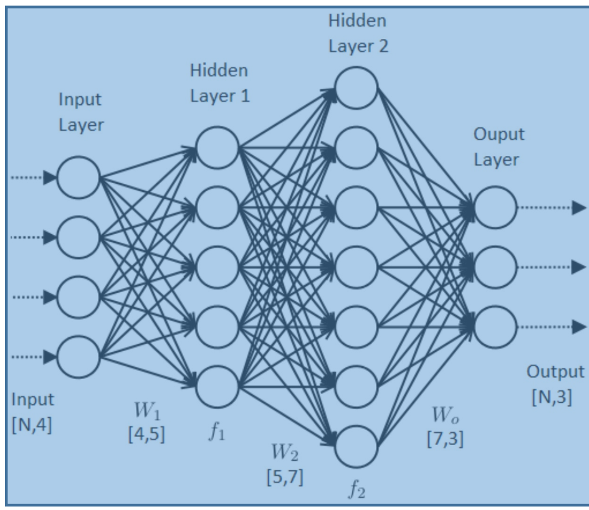
Actor

hurt leg



Actor

hurt leg



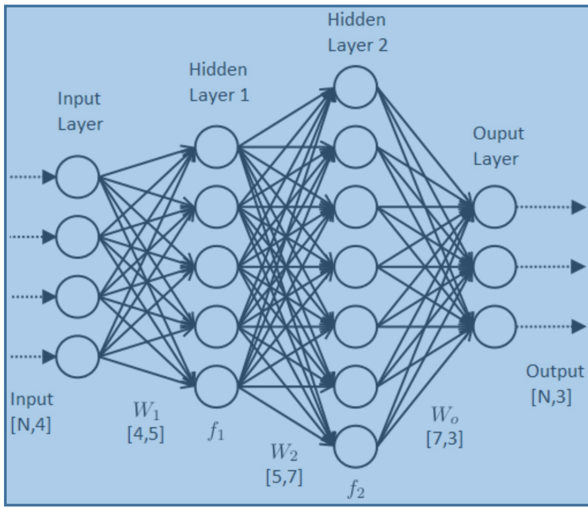
Actor

train hard

train less hard

give up

hurt leg



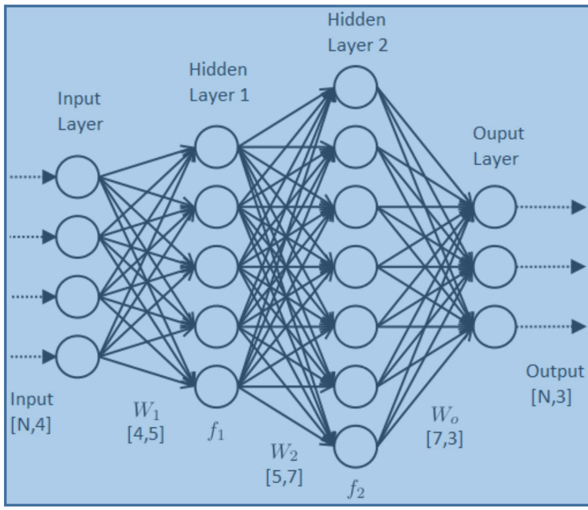
Actor

train hard
0.4

train less hard
0.3

give up
0.3

hurt leg



Actor

train hard
0.4

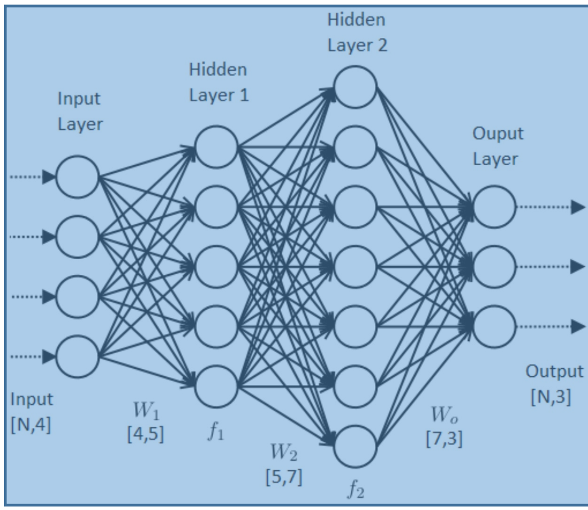
train less hard
0.3

give up
0.3

Two years later …


CLIVE BRUNSKILL/GETTY IMAGES SOUTH AMERICA/GETTY IMAGES
PUR
Rio2016

Now train the actor (neural network) …



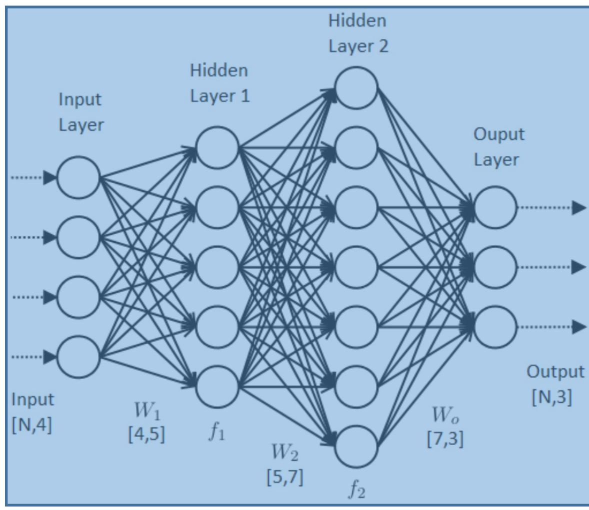
Actor

train hard
0.4

train less hard
0.3

give up
0.3

hurt leg



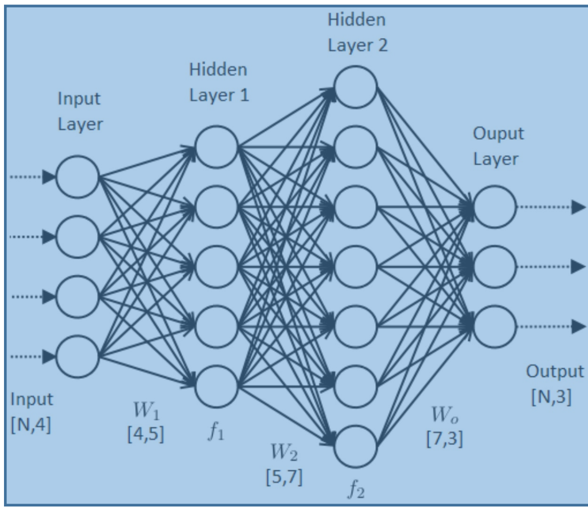
Actor

		Target output
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



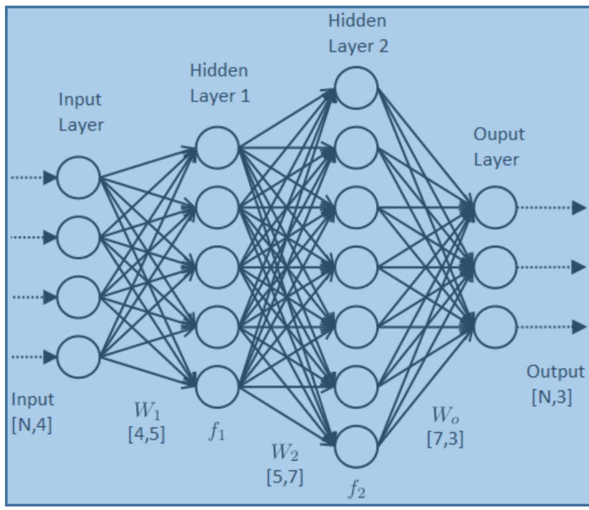
Actor

train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



Actor

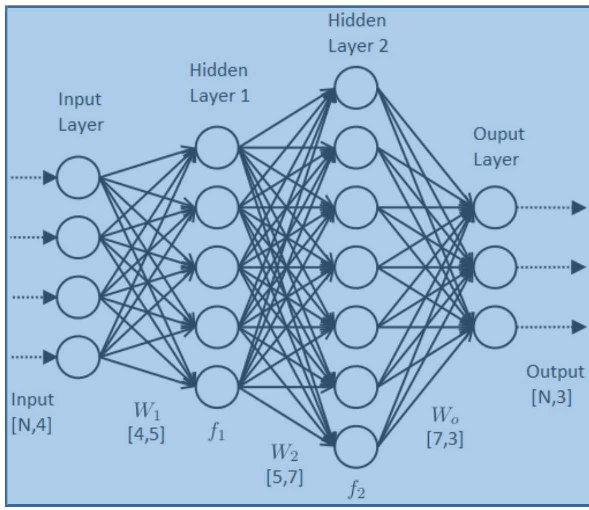
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Loss = cross-entropy between the two probability vectors

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



Actor

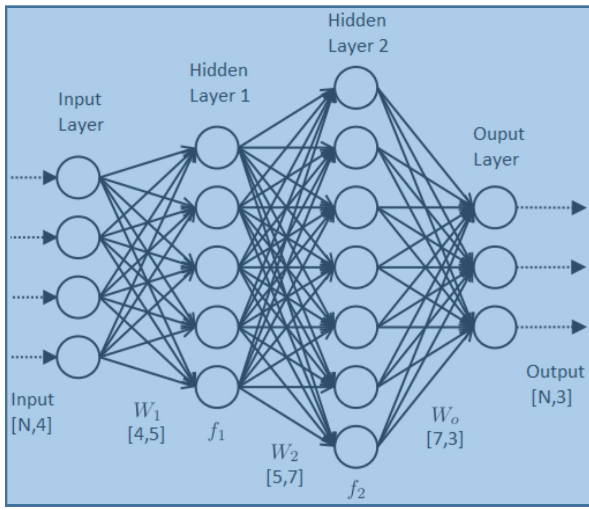
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Loss = cross-entropy between the two probability vectors X

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



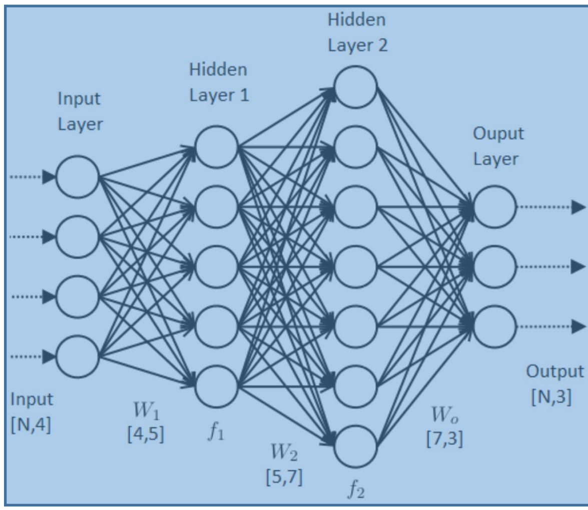
Actor

train hard	0.4 → 0.9	1
train less hard	0.3 → 0.06	0
give up	0.3 → 0.04	0

Loss = cross-entropy between the two probability vectors X

Another scenario…

hurt leg



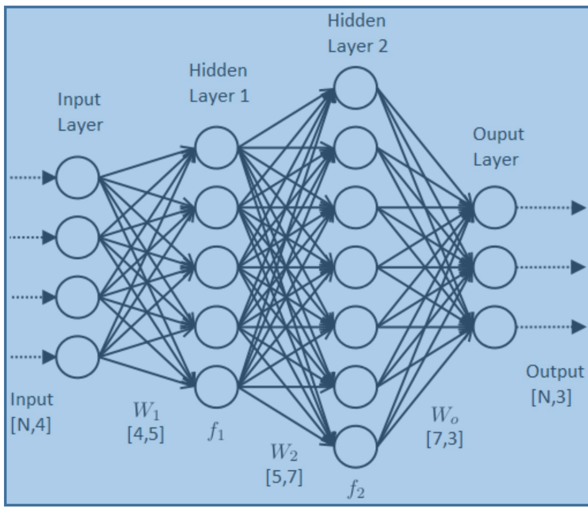
Actor

train hard
0.4

train less hard
0.3

give up
0.3

hurt leg



Actor

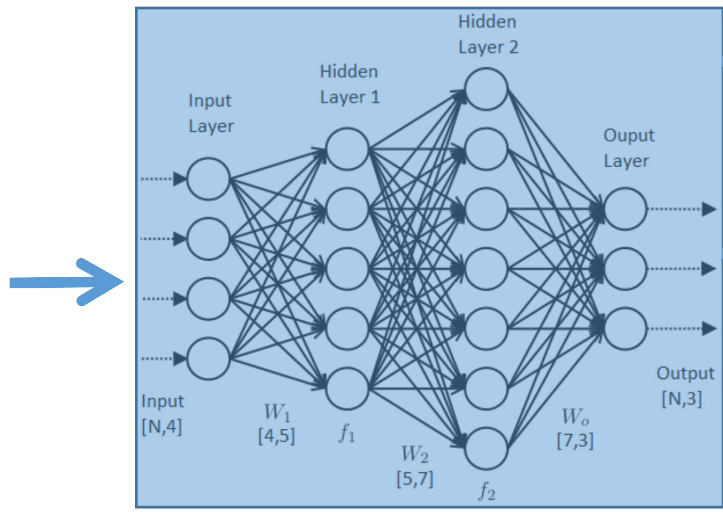
train hard
0.4

train less hard
0.3

give up
0.3

Two years later …

Now train the actor (neural network) …



train hard
0.4

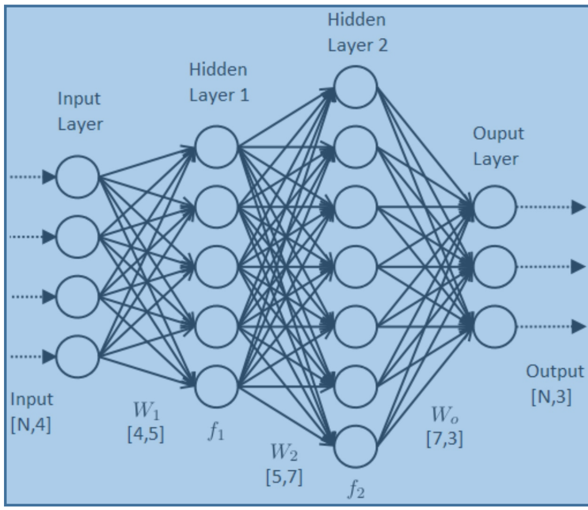
train less hard
0.3

give up
0.3

Actor

Target output

hurt leg



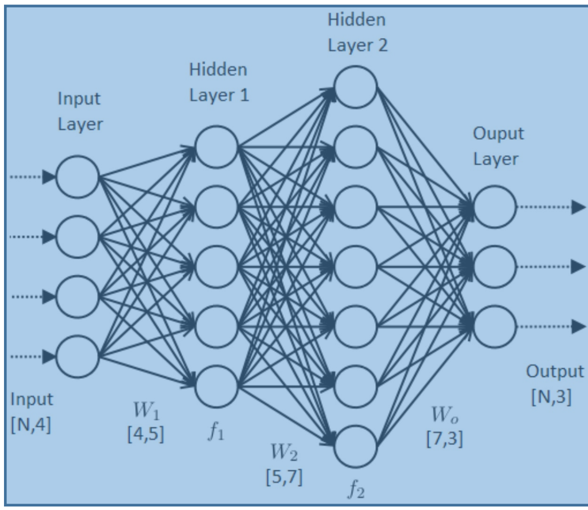
Actor

train hard 0.4	0
train less hard 0.3	0
give up 0.3	1

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



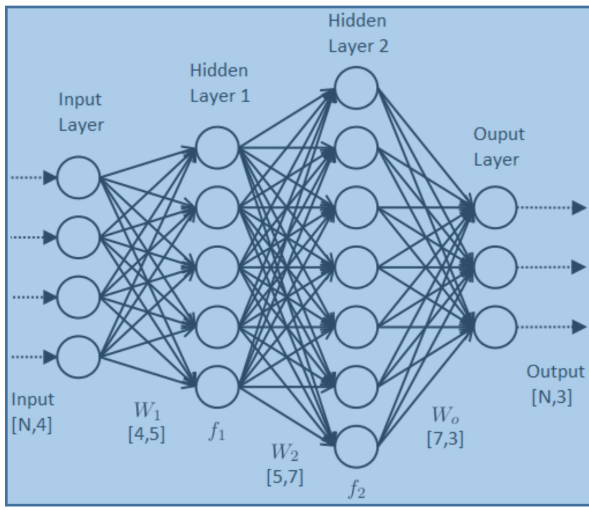
Actor

train hard 0.4	0
train less hard 0.3	0
give up 0.3	1

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



Actor

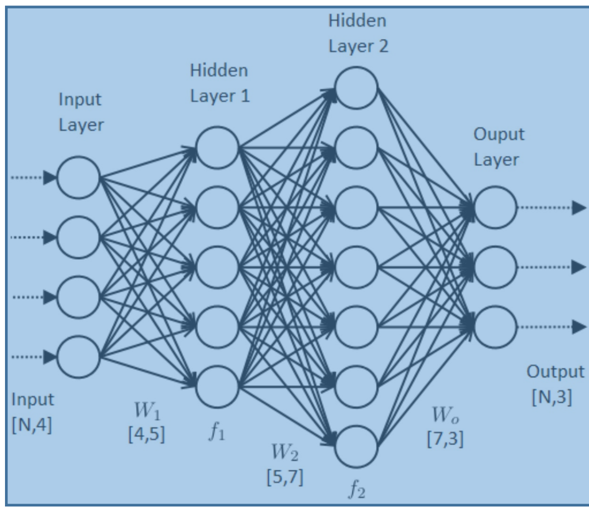
train hard	0.4	0
train less hard	0.3	0
give up	0.3	1

Loss = cross-entropy between the two probability vectors

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



Actor

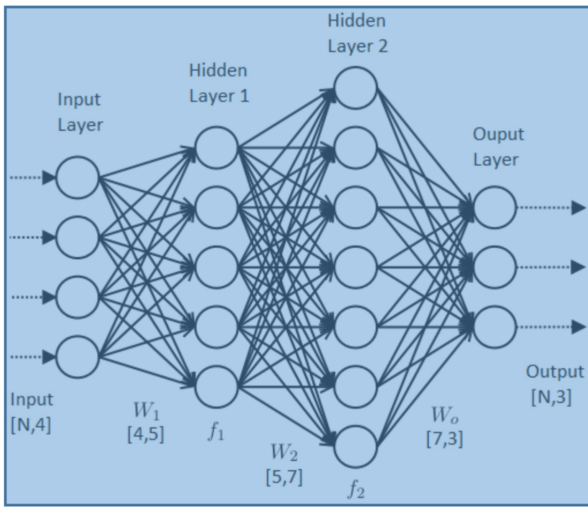
train hard	0.4	0
train less hard	0.3	0
give up	0.3	1

Loss = cross-entropy between the two probability vectors X

Tune weights to help the actor make the same choice with a higher probability.

Target output

hurt leg



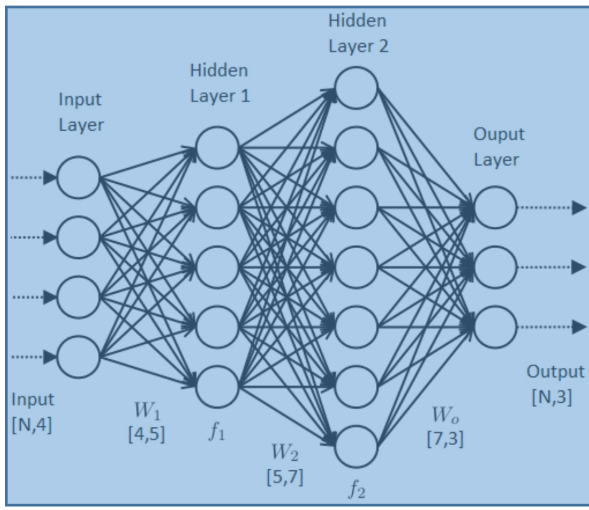
Actor

train hard	0.4 → 0.22	0
train less hard	0.3 → 0.28	0
give up	0.3 → 0.5	1

Loss = cross-entropy between the two probability vectors X

Another scenario…

hurt leg



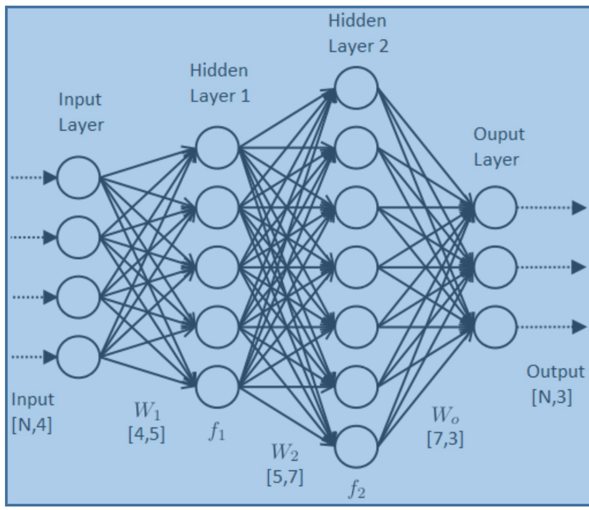
Actor

train hard
0.4

train less hard
0.3

give up
0.3

hurt leg



Actor

train hard

0.4

train less hard

0.3

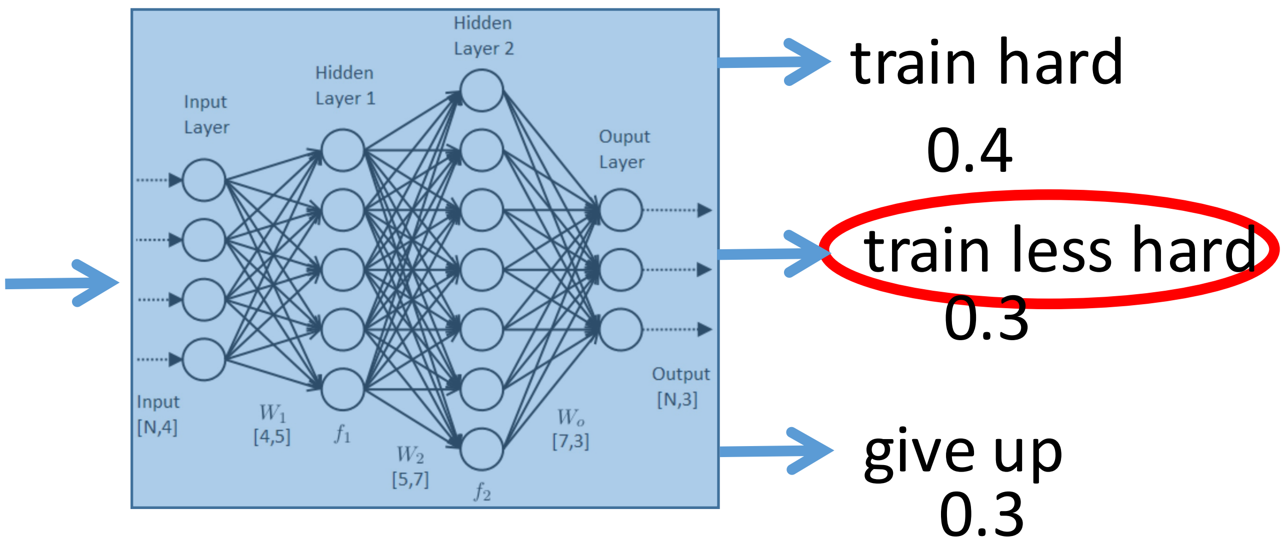
give up

0.3

Three years later …

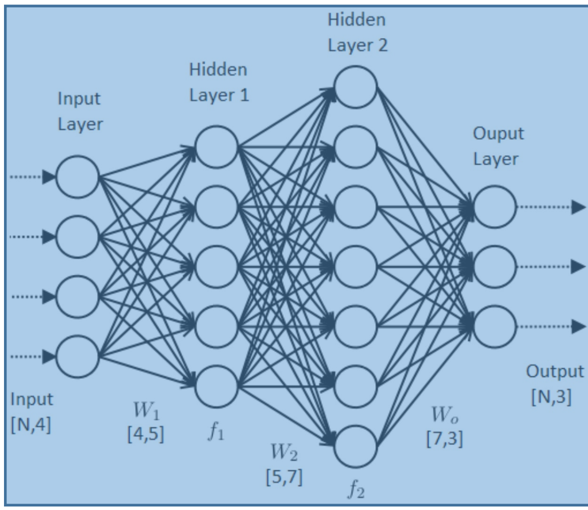

Lazy
Daze
TV
TONY

Now train the actor (neural network) …



Actor

hurt leg



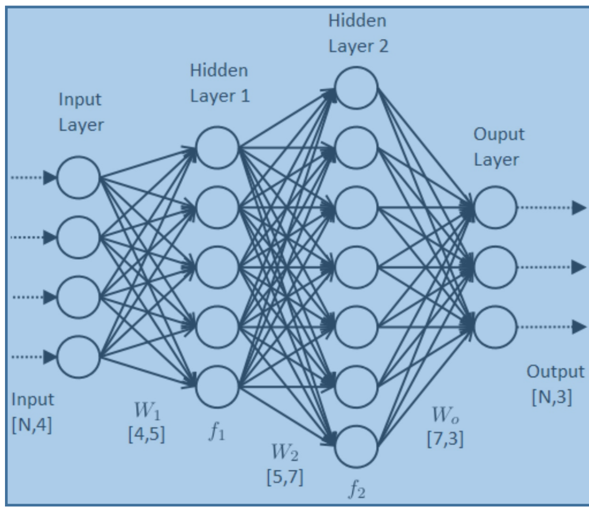
Actor

		Target output
train hard	0.4	0
train less hard	0.3	1
give up	0.3	0

Tune weights to help the actor make the same choice with a lower probability.

Target output

hurt leg



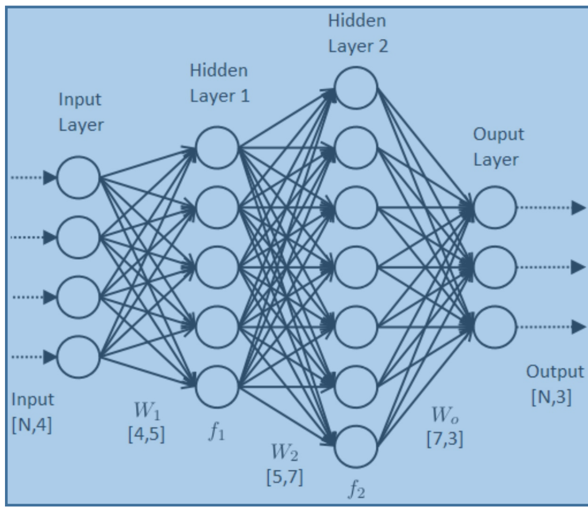
Actor

Action	Probability	Target output
train hard	0.4	0
train less hard	0.3	1
give up	0.3	0

Tune weights to help the actor make the same choice with a lower probability.

Target output

hurt leg



Actor

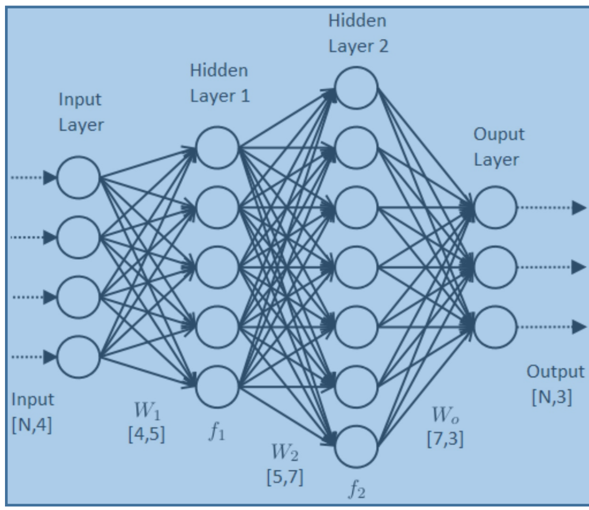
Action	Probability	Target output
train hard	0.4	0
train less hard	0.3	1
give up	0.3	0

Loss = cross-entropy between the two probability vectors

Tune weights to help the actor make the same choice with a lower probability.

Target output

hurt leg



Actor

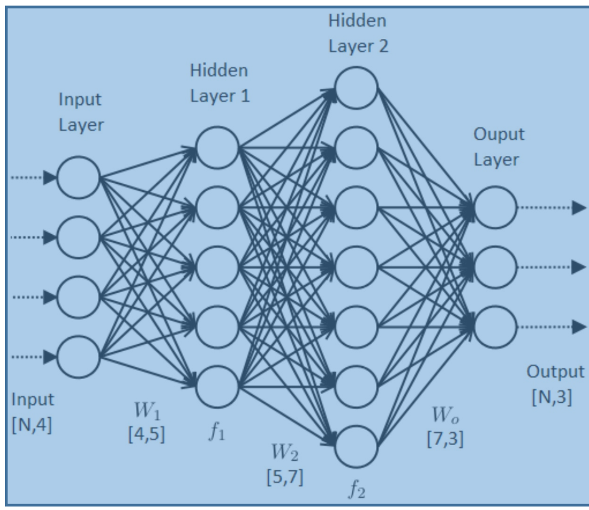
train hard	0.4	0
train less hard	0.3	1
give up	0.3	0

Loss = cross-entropy between the two probability vectors X

Tune weights to help the actor make the same choice with a lower probability.

Target output

hurt leg



Actor

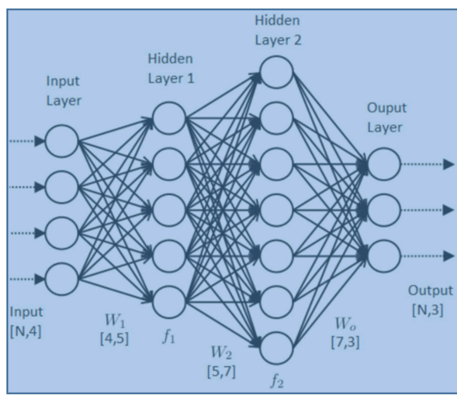
train hard	0.4 → 0.42	0
train less hard	0.3 → 0.2	1
give up	0.3 → 0.38	0

Loss = cross-entropy between the two probability vectors X

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n,\theta)$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla logp\underline{(a_t^n|s_t^n,\theta)}$$

hurt leg



Actor

train hard
0.4

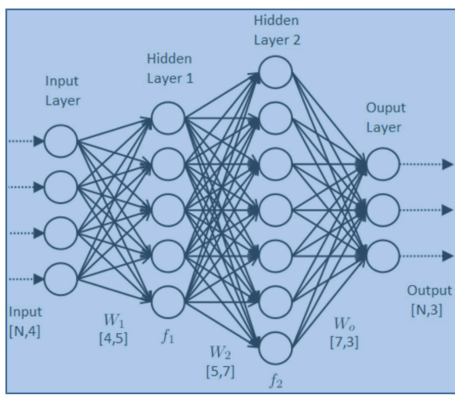
train less hard
0.3

give up
0.3

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\underline{\nabla \log p(a_t^n|s_t^n,\theta)}$$

negative
Cross entropy

hurt leg



Actor

		Target output
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Minimize Cross Entropy:

$$-\sum_{i=1}^{3} \hat{y}_i log y_i$$

$$-\sum_{i=1}^{3} \hat{y}_i log y_i$$

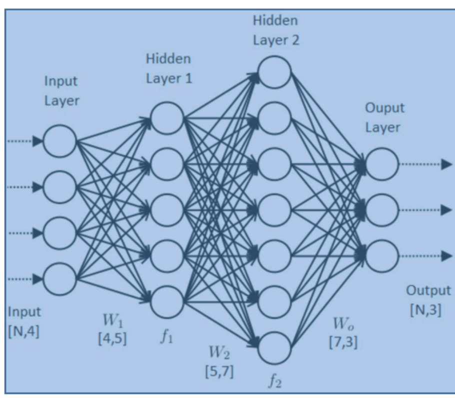
$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \underline{\nabla log p(a_t^n | s_t^n, \theta)}$$

negative

Cross entropy

Target output

hurt leg



Actor

train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

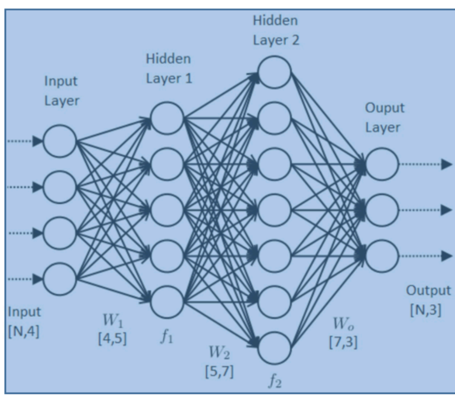
$$-\sum_{i=1}^{3} \hat{y}_i \log y_i$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \underline{\nabla \log p(a_t^n | s_t^n, \theta)}$$

negative

Cross entropy

hurt leg



Actor

		Target output
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

$$-\sum_{i=1}^{3} \hat{y}_i log y_i$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla \underline{log p(a_t^n|s_t^n,\theta)}$$

negative

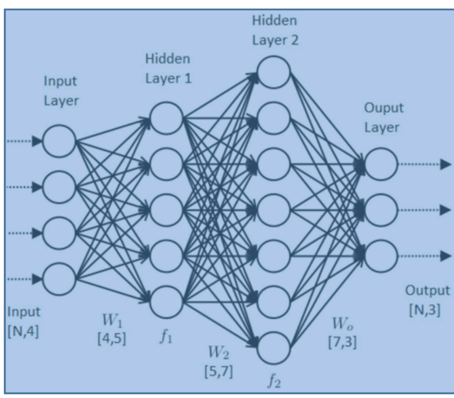
Cross

entropy

Target

output

hurt leg



Actor

train hard 0.4 | 1

train less hard 0.3 | 0

give up 0.3 | 0

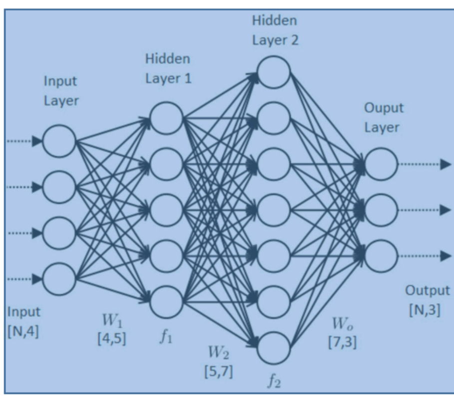
$$-\sum_{i=1}^{3} \hat{y}_i log y_i$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla log p(a_t^n | s_t^n, \theta)$$

negative

Cross entropy

hurt leg



Actor

train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

Target output

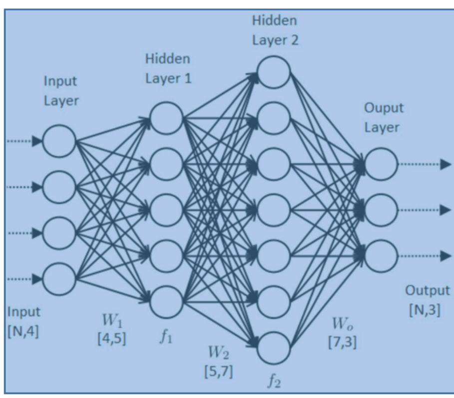
$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla logp(a_t^n|s_t^n,\theta)$$

$$-\sum_{i=1}^{3}\hat{y}_i log y_i$$

negative Cross entropy

Target output

hurt leg



Actor

train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

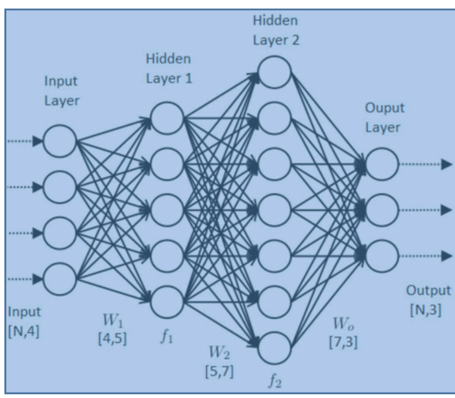
$$-\sum_{i=1}^{3} \hat{y}_i log y_i$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \underline{\nabla log p(a_t^n | s_t^n, \theta)}$$

negative
Cross
entropy

Target
output

hurt leg



Actor

train hard
0.4

train less hard
0.3

give up
0.3

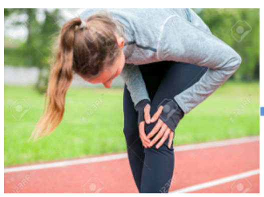
1

0

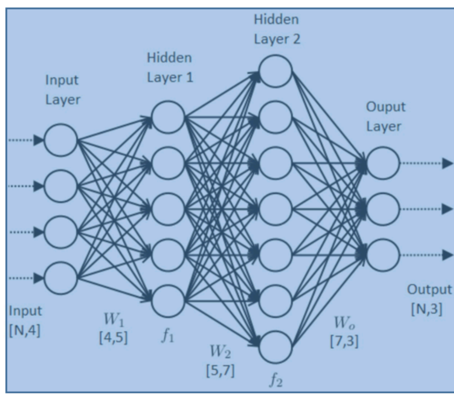
0

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\underline{(R(\tau^n) - b)}\nabla logp(a_t^n|s_t^n,\theta)$$

Constant weight for this input-output pair



hurt leg



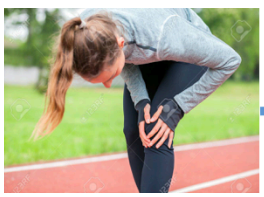
Actor

		Target output
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

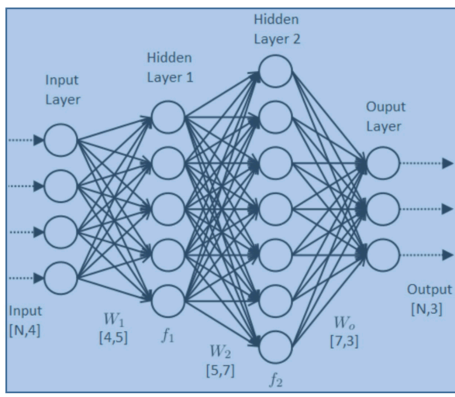
$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla log p(a_t^n|s_t^n,\theta)$$

Sum over all the input-output pairs

Target output



hurt leg



Actor

train hard 0.4 — 1

train less hard 0.3 — 0

give up 0.3 — 0

1. Use the current neural network (actor) to play the game, to get data from many episodes.

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n,\theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla \log p(a_t^n|s_t^n, \theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n, \theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n, \theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

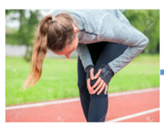
$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla logp(a_t^n|s_t^n, \theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

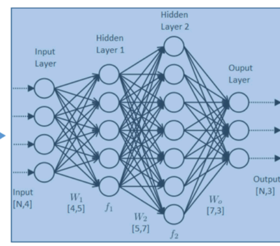
2. As a result, we get many triplets (observation, action, reward).

3. Turn the reinforcement learning problem into a classification problem. Train the network.

Sum over all the input-output pairs



hurt leg



Actor

Output	Probability	Target output
train hard	0.4	1
train less hard	0.3	0
give up	0.3	0

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla \log p(a_t^n|s_t^n,\theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

3. Turn the reinforcement learning problem into a classification problem. Train the network.

4. Use the new network to collect more data. Use the new data to train the new network.

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla log p(a_t^n|s_t^n, \theta)$$

1. Use the current neural network (actor) to play the game, to get data from many episodes.
2. As a result, we get many triplets (observation, action, reward).
3. Turn the reinforcement learning problem into a classification problem. Train the network.
4. Use the new network to collect more data. Use the new data to train the new network.
5. Repeat the above steps, until the network's performance converges.

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n) - b)\nabla log p(a_t^n|s_t^n, \theta)$$

The following lecture is based on the interesting lecture of Prof. Hung-yi Lee “Deep Reinforcement Learning”
https://www.youtube.com/watch?v=W8XF3ME8G2I&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=33

Policy Gradient

Policy Gradient

Given actor parameter θ

Policy Gradient

Given actor parameter θ

$$\tau^1: \quad \begin{array}{c} (s_1^1, a_1^1) \\ (s_2^1, a_2^1) \\ \vdots \end{array}$$

Policy Gradient

Given actor parameter θ

$$
\begin{array}{lll}
\tau^1: & (s_1^1, a_1^1) & R(\tau^1) \\
 & (s_2^1, a_2^1) & R(\tau^1) \\
 & \vdots & \vdots
\end{array}
$$

Policy Gradient

Given actor parameter θ

$$
\begin{array}{lll}
\tau^1: & (s_1^1, a_1^1) & R(\tau^1) \\
 & (s_2^1, a_2^1) & R(\tau^1) \\
 & \vdots & \vdots \\
\tau^2: & (s_1^2, a_1^2) & R(\tau^2) \\
 & (s_2^2, a_2^2) & R(\tau^2) \\
 & \vdots & \vdots
\end{array}
$$

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

$\vdots$ $\vdots$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla logp(a_t^n | s_t^n, \theta)$$

Data Collection

推出來的這個式子

Policy Gradient

Given actor parameter θ

$$
\begin{array}{lll}
\tau^1: & (s_1^1, a_1^1) & R(\tau^1) \\
 & (s_2^1, a_2^1) & R(\tau^1) \\
 & \vdots & \vdots \\
\tau^2: & (s_1^2, a_1^2) & R(\tau^2) \\
 & (s_2^2, a_2^2) & R(\tau^2) \\
 & \vdots & \vdots
\end{array}
$$

Update Model

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

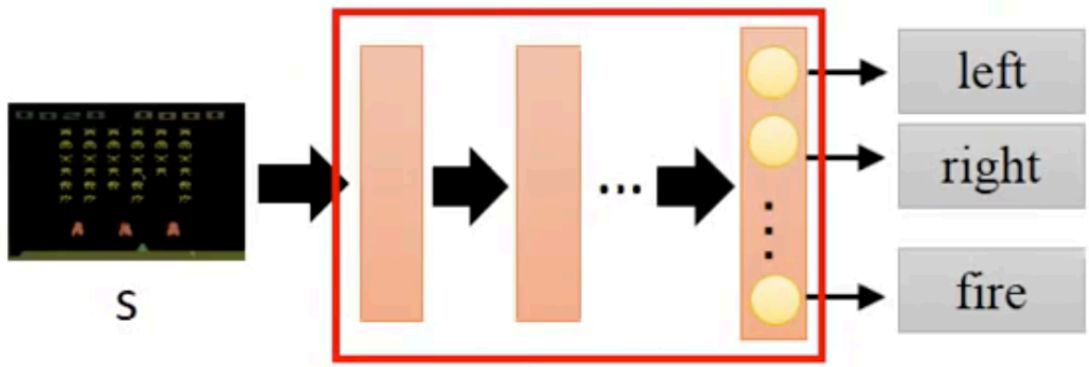
$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n | s_t^n, \theta)$$

Data Collection

你再去玩 N 次遊戲

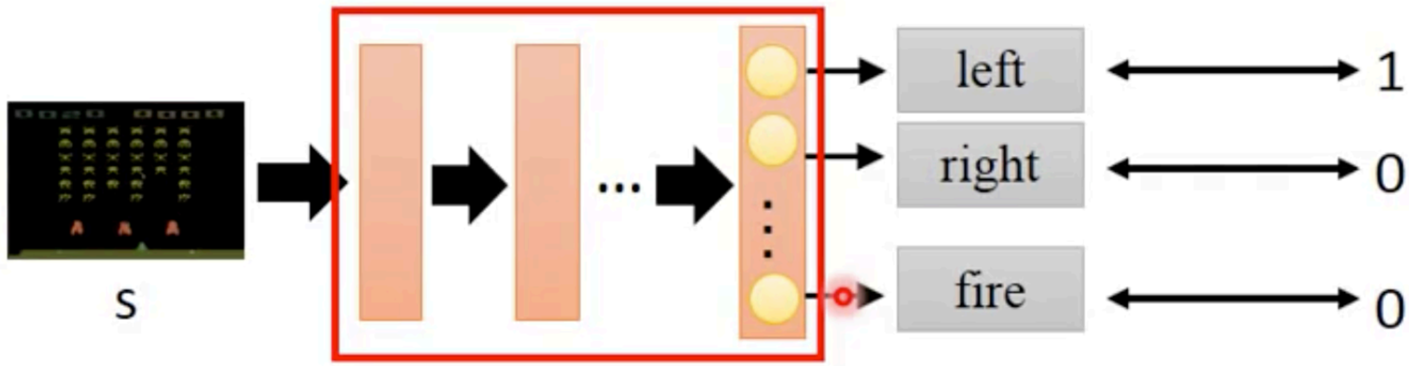
Policy Gradient

Considered as Classification Problem



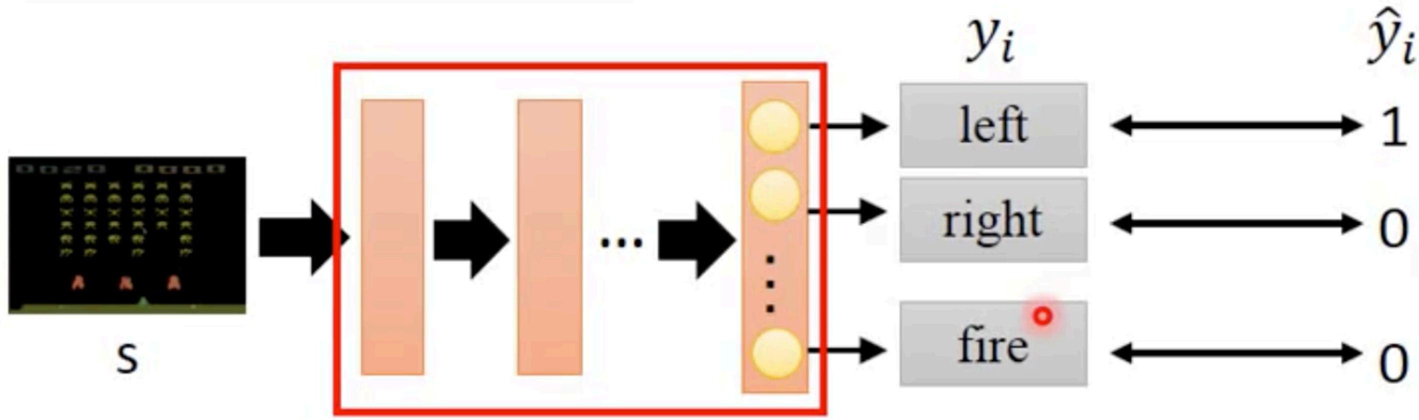
Policy Gradient

Considered as Classification Problem



Policy Gradient

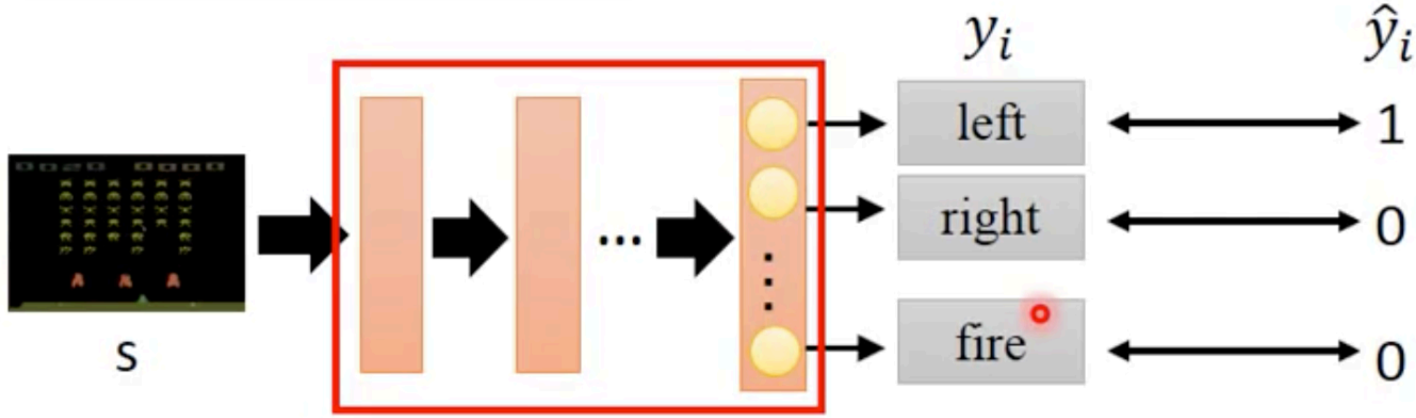
<u>Considered as Classification Problem</u>



Policy Gradient

<u>Considered as Classification Problem</u>

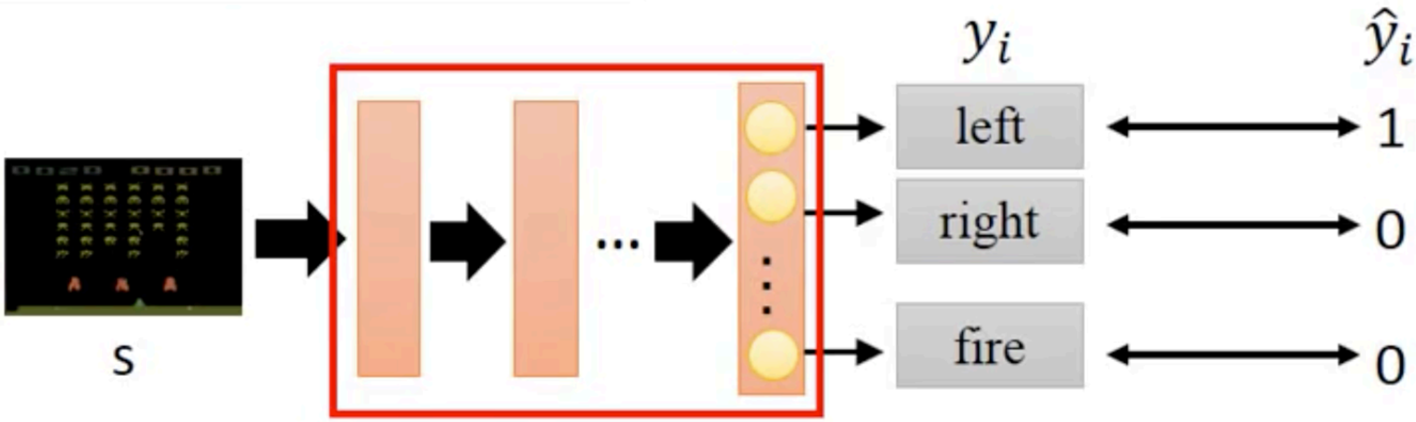
$$\text{Minimize: } -\sum_{i=1}^{3} \hat{y}_i log y_i$$



Policy Gradient

<u>Considered as Classification Problem</u>

Minimize: $-\sum_{i=1}^{3} \hat{y}_i log y_i$

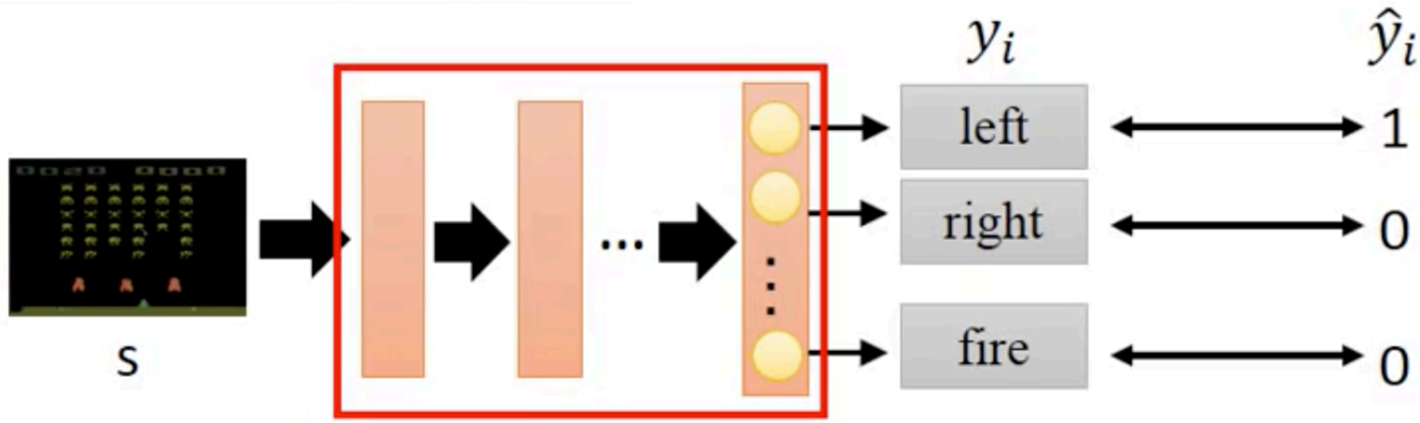


Maximize: $log y_i =$

Policy Gradient

Considered as Classification Problem

Minimize: $-\sum_{i=1}^{3} \hat{y}_i log y_i$



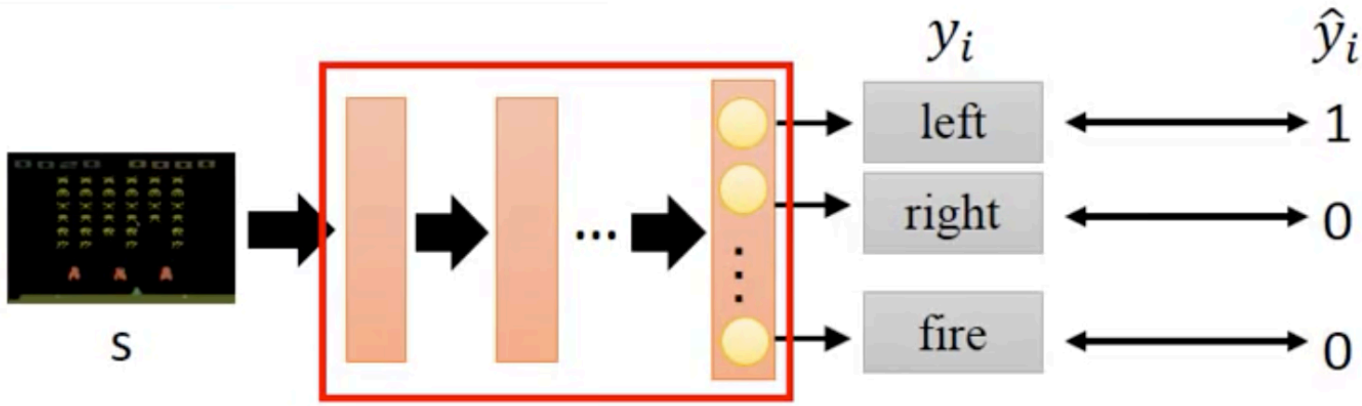
Maximize: $log y_i =$

$log P("left"|s)$

Policy Gradient

Considered as Classification Problem

Minimize: $-\sum_{i=1}^{3} \hat{y}_i log y_i$



Maximize: $log y_i =$

$log P("left"|s)$

$$\theta \leftarrow \theta + \eta \nabla log P("left"|s)$$

Policy Gradient

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \nabla \log p(a_t^n | s_t^n, \theta)$$

Given actor parameter θ

τ^1:	(s_1^1, a_1^1)	$R(\tau^1)$
	(s_2^1, a_2^1)	$R(\tau^1)$
	$\vdots$	$\vdots$
τ^2:	(s_1^2, a_1^2)	$R(\tau^2)$
	(s_2^2, a_2^2)	$R(\tau^2)$
	$\vdots$	$\vdots$

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

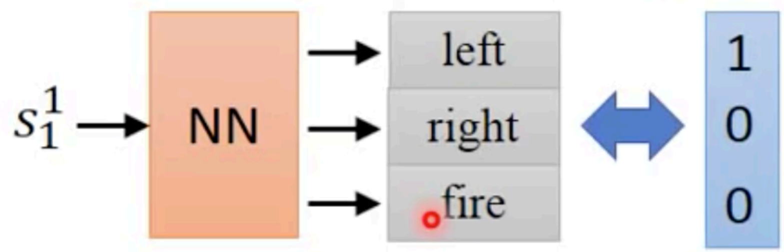
$\vdots$ $\vdots$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \boxed{\quad} \nabla logp(a_t^n | s_t^n, \theta)$$

$$a_1^1 = left$$



Policy Gradient

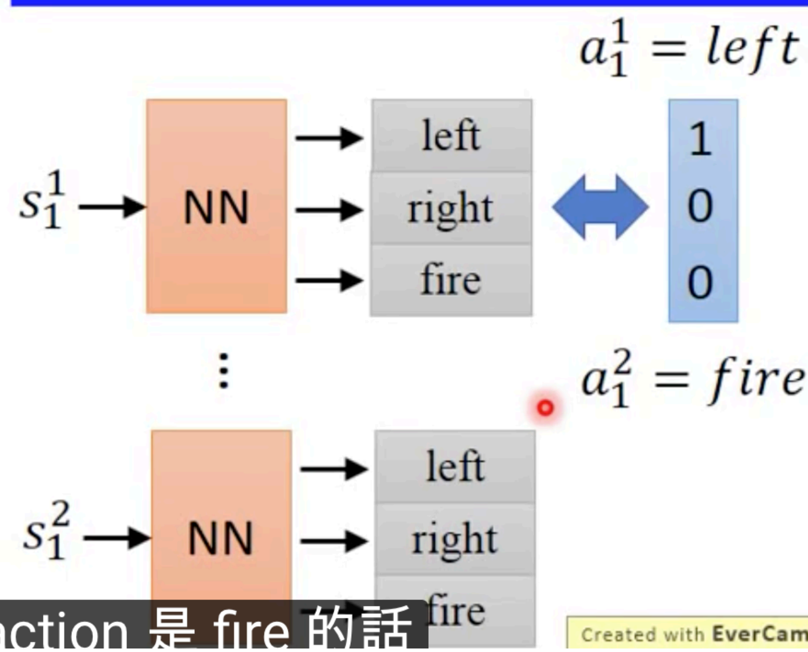
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \nabla \log p(a_t^n | s_t^n, \theta)$$

Given actor parameter θ

τ^1:	(s_1^1, a_1^1)	$R(\tau^1)$
	(s_2^1, a_2^1)	$R(\tau^1)$
	$\vdots$	$\vdots$
τ^2:	(s_1^2, a_1^2)	$R(\tau^2)$
	(s_2^2, a_2^2)	$R(\tau^2)$
	$\vdots$	$\vdots$



我們採取的 action 是 fire 的話

Policy Gradient

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \boxed{} \nabla logp(a_t^n|s_t^n,\theta)$$

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

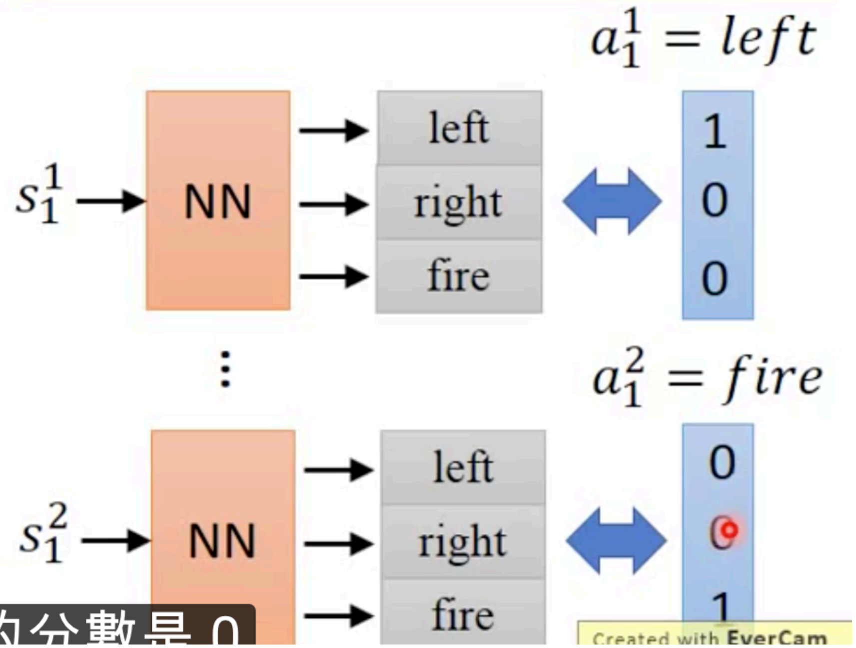
(s_2^1, a_2^1) $R(\tau^1)$

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

⋮ ⋮



其他的分數是 0

Policy Gradient

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta =$$

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n | s_t^n, \theta)$$

Given actor parameter θ

$$\begin{array}{lll} \tau^1: & (s_1^1, a_1^1) & R(\tau^1) \\ & (s_2^1, a_2^1) & R(\tau^1) \\ & \vdots & \vdots \\ \tau^2: & (s_1^2, a_1^2) & R(\tau^2) \\ & (s_2^2, a_2^2) & R(\tau^2) \\ & \vdots & \vdots \end{array}$$

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

$\vdots$ $\vdots$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n | s_t^n, \theta)$$

Each training data is weighted by $R(\tau^n)$

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$ **2**

(s_2^1, a_2^1) $R(\tau^1)$ **2**

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$ **1**

(s_2^2, a_2^2) $R(\tau^2)$ **1**

⋮ ⋮

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n | s_t^n, \theta)$$

Each training data is weighted by $R(\tau^n)$

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$ **2**

(s_2^1, a_2^1) $R(\tau^1)$ **2**

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$ **1**

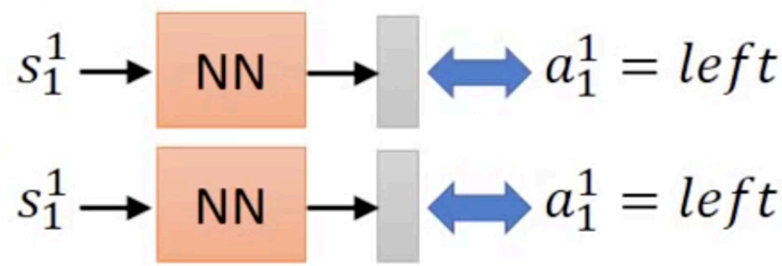
(s_2^2, a_2^2) $R(\tau^2)$ **1**

⋮ ⋮

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla logp(a_t^n | s_t^n, \theta)$$

Each training data is weighted by $R(\tau^n)$



Policy Gradient

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla \log p(a_t^n | s_t^n, \theta)$$

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$ 2

(s_2^1, a_2^1) $R(\tau^1)$ 2

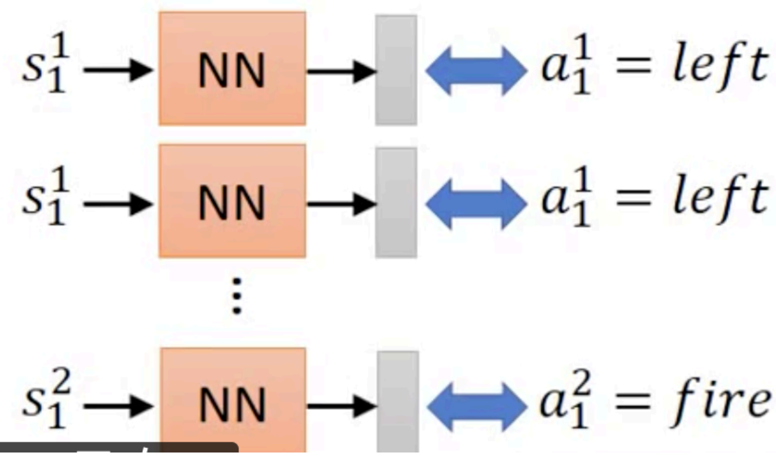
⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$ 1

(s_2^2, a_2^2) $R(\tau^2)$ 1

⋮ ⋮

Each training data is weighted by $R(\tau^n)$



1. Use the current neural network (actor) to play the game, to get data from many episodes.

2. As a result, we get many triplets (observation, action, reward).

3. Turn the reinforcement learning problem into a classification problem. Train the network.

4. Use the new network to collect more data. Use the new data to train the new network.

5. Repeat the above steps, until the network's performance converges.

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla \log p(a_t^n | s_t^n, \theta)$$