Situated Language Learning with Policy Gradients

L14. Felix Hill

DeepMind
This feels like a long shot....

"Ahem....don't you think this bedroom is a bit of a pigsty?"

???????????
Reinforcement learning for games

- General-purpose learning algorithm
- Works for many different problems
- Teaches us about the game

Mnih et al, 2015
Supervised learning

\[ y = [0, 0, 0, 1, 0, 0] \]

A neural network
Supervised learning

Find weights $\theta$ to minimize e.g.

$$\sum_{x,y \in D} - \log(g(x, y, \theta))$$

where $g(x, y, \theta) = M(x, \theta)|_y = P(y|x, \theta)$

A dataset $D$ of $(x, y)$ input-output pairs

$y = [0, 0, 0, 1, 0, 0]$
Supervised learning

Find theta to minimize e.g.

\[
\sum_{x, y \in D} - \log(g(x, y, \theta)) = C(x, y, \theta)
\]

A dataset D of (x, y) input-output pairs

where

\[
g(x, y, \theta) = M(x, \theta) |_{y} = P(y|x, \theta)
\]

y = [0, 0, 0, 1, 0, 0]

\[
M(x, \theta)
\]

\[
\theta \rightarrow \theta + \alpha \nabla_{\theta} C(x, y, \theta)
\]

Improve M (\theta values) by gradient descent

The "cost" function
Supervised learning

\[ y = [0, 0, 0, 1, 0, 0] \]

\[ M(x, \theta) \]

Reinforcement learning

\[ r_1 = 0 \quad r_2 = 0 \quad r_3 = 1 \]

\[ \pi(x_1, \theta) \quad \pi(x_2, \theta) \quad \pi(x_3, \theta) \]

\[ \text{UP} \quad \text{RIGHT} \quad \text{PICK} \]

\[ [0.12, \textbf{0.64}, 0.07, 0.21] \quad [0.03, 0.24, \textbf{0.47}, 0.22] \quad [\textbf{0.92}, 0.14, 0.27, 0.11] \]
Reinforcement learning

\[ \pi(x_1, \theta) \]

\[ \begin{bmatrix} 0.12, & 0.64, & 0.07, & 0.21 \end{bmatrix} \]

\[ \pi(x_2, \theta) \]

\[ \begin{bmatrix} 0.03, & 0.24, & 0.47, & 0.22 \end{bmatrix} \]

\[ \pi(x_3, \theta) \]

\[ \begin{bmatrix} 0.92, & 0.14, & 0.27, & 0.11 \end{bmatrix} \]

\[ r_1 = 0 \]
\[ r_2 = 0 \]
\[ r_3 = 1 \]

An environment that gives us observations \( x \)
Reinforcement learning

And scalar rewards $r$

$\pi(x_1, \theta)$

$\pi(x_2, \theta)$

$\pi(x_3, \theta)$

$\begin{bmatrix} 0.12, 0.64, 0.07, 0.21 \end{bmatrix}$

$\begin{bmatrix} 0.03, 0.24, 0.47, 0.22 \end{bmatrix}$

$\begin{bmatrix} 0.92, 0.14, 0.27, 0.11 \end{bmatrix}$

$\begin{align*}
  r_1 &= 0 \\
  r_2 &= 0 \\
  r_3 &= 1
\end{align*}$

An environment that gives us observations $x$

$x_1$

$1$

$x_2$

$2$

$x_3$

$3$
And scalar rewards $r$

$r_1 = 0 \quad r_2 = 0 \quad r_3 = 1$

The "policy" $\pi$ predicts action probabilities given observations - Like the supervised $M$

An environment that gives us observations $x$
Reinforcement learning

And scalar rewards $r$

$\pi(x_1, \theta)$

$\pi(x_2, \theta)$

$\pi(x_3, \theta)$

$\begin{bmatrix} 0.12, & 0.64, & 0.07, & 0.21 \end{bmatrix}$

$\begin{bmatrix} 0.03, & 0.24, & 0.47, & 0.22 \end{bmatrix}$

$\begin{bmatrix} 0.92, & 0.14, & 0.27, & 0.11 \end{bmatrix}$

$\begin{bmatrix} \text{UP} \end{bmatrix}$

$\begin{bmatrix} \text{RIGHT} \end{bmatrix}$

$\begin{bmatrix} \text{PICK} \end{bmatrix}$

$r_1 = 0$

$r_2 = 0$

$r_3 = 1$

Only rewards to learn from - no guarantee 'actions' are correct

An environment that gives us observations $x$

$\begin{bmatrix} x_1 \end{bmatrix}$

$\begin{bmatrix} x_2 \end{bmatrix}$

$\begin{bmatrix} x_3 \end{bmatrix}$

The "policy" $\pi$ predicts actions given observations

$\begin{bmatrix} 1 \end{bmatrix}$

$\begin{bmatrix} 2 \end{bmatrix}$

$\begin{bmatrix} 3 \end{bmatrix}$

Only rewards to learn from - no guarantee 'actions' are correct
Gradient methods for Reinforcement Learning?

We want to optimise

\[ J(\theta) = \mathbb{E}_{\pi(\theta)}(R) \]

where

\[ R = \sum_{t=1}^{k} \gamma^t r_t \]

Find the policy weights that give us the highest expected return.

All the reward I got from the environment.
Gradient methods for Reinforcement Learning?

We want to optimise

$$J(\theta) = \mathbb{E}_{\pi(\theta)}(R)$$

where

$$R = \sum_{t=1}^{k} \gamma^t r_t$$

If we knew $\nabla_\theta J(\theta)$

we could just do gradient ascent!

$$\theta \rightarrow \theta + \alpha \nabla_\theta J(\theta)$$
Gradient methods for Reinforcement Learning?

We want to optimise

$$J(\theta) = \mathbb{E}_{\pi(\theta)}(R)$$

where

$$R = \sum_{t=1}^{k} \gamma^t r_t$$

Trajectory

$$\tau = \{s_1 \cdots s_k, \}$$
Gradient methods for Reinforcement Learning?

We want to optimise

\[ J(\theta) = \mathbb{E}_{\pi(\theta)}(R) \]

where

\[ R = \sum_{t=1}^{k} \gamma^t r_t \]

Trajectory

\[ \mathcal{T} = \{s_1, \ldots, s_k, \} \]

Return

\[ R(\mathcal{T}) = \sum_{t=1}^{k} \gamma^t r_t \]

For a stationary environment:

A given trajectory -> unique well-defined return
Gradient methods for Reinforcement Learning?

We want to optimise

\[ J(\theta) = \mathbb{E}_{\pi(\theta)}(R) \]

where

\[ R = \sum_{t=1}^{k} \gamma^t r_t \]

Trajectory

\[ \tau = \{ s_1, \ldots, s_k \} \]

Return

\[ R(\tau) = \sum_{t=1}^{k} \gamma^t r_t \]

So condition on trajectories

\[ J(\theta) = \mathbb{E}_{\pi(\theta)}(R) = \mathbb{E}_{\tau \in T|\pi}(R(\tau)) \]
Estimating a policy gradient in an environment

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \mid \pi(\theta)}(R(\tau))$$
Estimating a policy gradient in an environment

\[ \nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau|\pi(\theta)}(R(\tau)) \]

\[ = \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau|\pi_\theta) \]

*Definition of expectation*

*Space of all possible trajectories*
Estimating a policy gradient in an environment

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau|\pi(\theta)}(R(\tau))$$

$$= \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau|\pi_\theta) \quad \text{Definition of expectation}$$

$$= \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau|\pi_\theta) \quad \text{Only } P \text{ depends on } \theta$$
Estimating a policy gradient in an environment

\[ \nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau|\pi(\theta)}(R(\tau)) \]

\[ = \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau|\pi_\theta) \quad \text{Definition of expectation} \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau|\pi_\theta) \quad \text{Only } P \text{ depends on } \theta \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau|\pi_\theta) \frac{P(\tau|\pi_\theta)}{P(\tau|\pi_\theta)} \times 1 \]


Estimating a policy gradient in an environment

\[ \nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_\tau | \pi(\theta)(R(\tau)) \]

\[ = \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau | \pi_\theta) \quad \text{Definition of expectation} \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau | \pi_\theta) \quad \text{Only } P \text{ depends on } \theta \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau | \pi_\theta) \frac{P(\tau | \pi_\theta)}{P(\tau | \pi_\theta)} \times 1 \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta \log P(\tau | \pi_\theta) P(\tau | \pi_\theta) \quad \text{By chain rule} \quad \nabla_\theta \log P(\tau | \pi_\theta) \leftrightarrow \frac{\nabla_\theta P(\tau | \pi_\theta)}{P(\tau | \pi_\theta)} \]
Estimating a policy gradient in an environment

\[ \nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau | \pi(\theta)} (R(\tau)) \]

\[ = \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau | \pi_\theta) \quad \text{Definition of expectation} \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau | \pi_\theta) \quad \text{Only } P \text{ depends on } \theta \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau | \pi_\theta) \frac{P(\tau | \pi_\theta)}{P(\tau | \pi_\theta)} \times 1 \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta \log P(\tau | \pi_\theta) P(\tau | \pi_\theta) \quad \text{By chain rule} \quad \nabla_\theta \log P(\tau | \pi_\theta) \rightarrow \frac{\nabla\theta P(\tau | \pi_\theta)}{P(\tau | \pi_\theta)} \]

\[ = \mathbb{E}_{\tau \in T} (R(\tau) \nabla_\theta \log P(\tau | \pi_\theta)) \quad \text{Definition of expectation} \]
Estimating a policy gradient in an environment

\[ \nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_\tau | \pi(\theta) (R(\tau)) \]

\[ = \nabla_\theta \sum_{\tau \in T} R(\tau) P(\tau | \pi_\theta) \quad \text{Definition of expectation} \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta P(\tau | \pi_\theta) \quad \text{Only } P \text{ depends on } \theta \]

\[ = \sum_{\tau \in T} R(\tau) \nabla_\theta \log P(\tau | \pi_\theta) P(\tau | \pi_\theta) \quad \text{By chain rule} \]

\[ = \mathbb{E}_{\tau \in T} \left( R(\tau) \nabla_\theta \log P(\tau | \pi_\theta) \right) \quad \text{Definition of expectation} \]

The gradient of the objective wrt. policy weights

A quantity that I can compute by following a trajectory \( \tau \)
The REINFORCE algorithm

To estimate $\nabla_\theta J(\theta)$ the gradient of our objective wrt. the parameters of the policy function.

We can estimate

$$\mathbb{E}_{\tau \in T}(R(\tau) \nabla_\theta \log P(\tau|\pi_\theta))$$

Notice also that

$$\nabla_\theta \log(P(\tau|\pi)) = \nabla_\theta \sum_{t=1}^{k} \log P(a_t|\pi, s_t) = \sum_{t=1}^{k} \nabla_\theta \log P(a_t|\pi, s_t)$$

So - **act in the environment** (follow trajectories).

- At each time step, remember: $\nabla_\theta \log P(a_t|\pi, s_t)$
- After each trajectory (episode), compute: $R(\tau) = \sum_{t=1}^{k} \gamma^t r_t$
The REINFORCE algorithm

Initialise $\theta$ randomly:

For episodes $\{s_1, a_1, r_1 \ldots s_k, a_k, r_k\} \sim \pi_\theta$

Compute $R(\tau) = \sum_{t=1}^{k} \gamma^t r_t$

For $t = 1 \ldots T$:

$$\theta \rightarrow \theta + \alpha R \nabla_\theta \log \pi(x_t, a_t)$$

The gradient of the policy network, evaluated at a particular input / output pair

If action choices led to good rewards, move weights to follow gradient (scaled by $R$)
Supervised learning

- **Something went well** (high LL) AND you know how to make it even better

- **Something went badly** (low LL) - you know how to fix it

Reinforcement learning

- **Something went well** (high R), you know what you did - can reinforce
  - Not sure if you could have done better

- **Something went badly** (low R) - no idea what you should have done
Learning language in RL environments

Language can refer to the visual **world**
- Similar to image captioning / VQA

Language can refer to **actions** and / or **policies**
- Like a lot of natural language does!

Where does **reward** come from?
DeepMind Lab

DeepMind Lab

Observations

Rewards

Pixels

"language"

Agent

Actions

jump

rotate up/down

rotate left/right

forward/back

strafe left/right

crouch

Deep RL alone (A3C) not enough
Start off small (or large)...

![Image of a yellow object in a green wall with circuit patterns, numbered '100' and '123' on the floor]
Colour words...
Shape words...
Language in DeepMind Lab: The Lexicon

Shapes (40) tv, ball, balloon, cake, can, cassette, chair, guitar, hairbrush, hat, ice lolly, ladder, mug, pencil, suitcase, toothbrush, key, bottle, car, cherries, fork, fridge, hammer, knife, spoon, apple, banana, cow, flower, jug, pig, pincer, plant, saxophone, shoe, tennis racket, tomato, tree, wine glass, zebra.

Colours (13) red, blue, white, grey, cyan, pink, orange, black, green, magenta, brown, purple, yellow.

Patterns (9) plain, chequered, crosses, stripes, discs, hex, pinstripe, spots, swirls.

Shades (3) light, dark, neutral.

Sizes (3) small, large, medium.

Auxiliary objectives
Unsupervised learning makes word learning possible

A 2-layer LSTM

V 3-layer CNN

a_t

Val_t

M 1-layer LSTM

tAE

LP

VR

"ladder"

Average Reward per Episode

Training Episodes
And provides insight into agents' 'thoughts'.

pick the chequered hair_brush
Combining exploration and language

Top-down view of the level

![Diagram showing a top-down view of a level with a character standing in front of a graph. The graph plots training episodes against a metric with a line indicating performance over time.]
Curriculum is critical

1. single-room layout
   - two object words and room descriptors

2. two room layout
   - two object words and room descriptors

---

Graphs showing performance over training episodes:

- Agent trained from scratch
- Agent previously trained on level 1
- Agent trained from scratch
- Agent previously trained on level 2
- Agent trained from scratch
- Agent previously trained on level 3
Curriculum is critical

1. single-room layout
   - two object words and room descriptors

2. two room layout
   - two object words and room descriptors

3. two room layout
   - medium object word / room descriptor vocabulary

Agent trained from scratch
Agent previously trained on level 1
Agent trained from scratch
Agent previously trained on level 2
Agent previously trained on level 1
Agent trained from scratch
Curriculum is critical

1. single-room layout
   - two object words and room descriptors

2. two room layout
   - two object words and room descriptors

3. two room layout
   - medium object word / room descriptor vocabulary

4. two room layout
   - full object word / room descriptor vocabulary

Agent trained from scratch
Agent previously trained on level 1
Agent trained from scratch
Agent previously trained on level 2
Agent previously trained on level 1
Agent trained from scratch
Agent previously trained on level 3
Agent previously trained on level 2
Agent previously trained on level 1
Agent trained from scratch
Isn't this all a bit convoluted?
Agents naturally generalise word composition...

Training

"ladder"
"mug"
"pencil"
"suitcase"
"toothbrush"

Test

"pink ladder"
"yellow mug"
"green pencil"

"red"
"green"
"blue"
"pink"

"red ladder"
"green mug"
"blue pencil"

Performance on training set
Performance on test set
Decompose before re-compose

Performance on training set
Performance on test set
Apply modifiers and predicates to novel objects

Training
"larger" (ball)
"smaller" (ball)
"larger" (mug)
"smaller" (mug)

Test
"larger" (pencil)
"smaller" (pencil)
Generalisation (zero-shot etc...)

AI researcher

**Environment**

**Agent**

lang + world

lang + world
Isn't learning slow?
Word learning gets quicker the more the agent 'knows'
Much like little people

**Figure 1.** Vocabulary development in two Danish children: the plateau period is followed by a period of accelerated growth referred to as the vocabulary spurt.
Much like little people

Figure 1. Vocabulary development in two Danish children: the plateau period is followed by a period of accelerated growth referred to as the vocabulary spurt.
Much like little people

Agent that already knows 20 words outside of training set

Agent that already knows 2 words outside of training set

Agent trained from scratch

conceptual 'bootstrapping'

vocabulary 'spurt'
How does the agent represent its knowledge?
Layerwise attention
Layerwise attention
Processing colour words

Agent visual input

Attention distribution

Language channel instruction

Visualise attention layer 3

Visualise attention layer 2

Visualise attention layer 1
Processing shape words
Conclusions

- Using RL we can ground language in **vision**, **actions** and **policies**
- Neural network policies enable natural **generalisation** and **composition**
- Ongoing work to scale approaches to **full sentences** and **natural commands**
  - What aspects of language are not covered by this approach?
  - What challenges do we face extending this?