Deep Learning and Reinforcement Learning

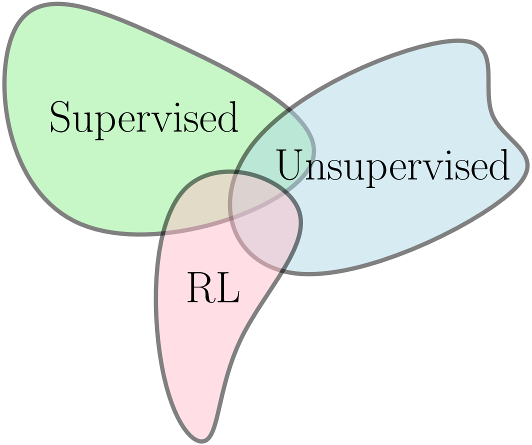
Razvan Pascanu (Google DeepMind)

Disclaimers:

Slides based on David Silver's Lecture Notes

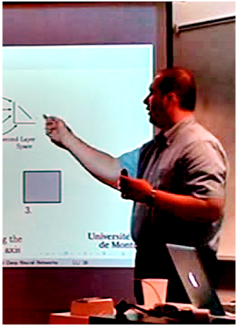
- From a DL perspective
- Not complete, but rather biased and focused
- It is meant to **make you want to learn this**

What is Reinforcement Learning ?

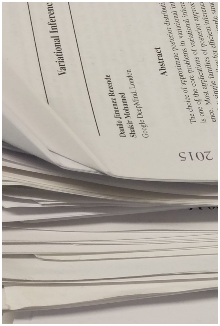


What is Reinforcement Learning ?

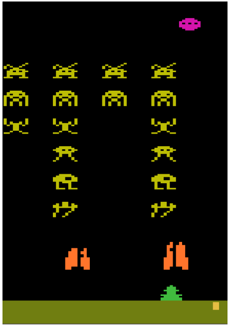
Supervised Learning



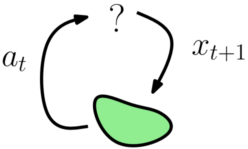
Unsupervised Learning



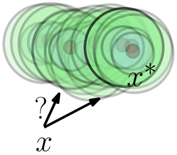
Reinforcement Learning



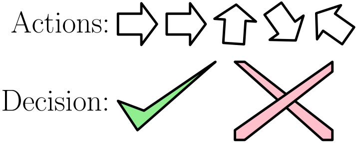
Laundry list of differences for RL



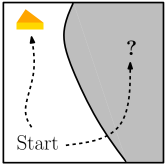
Active learning



Moving target

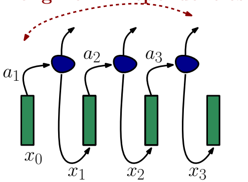


Weak error signal



Exploration/Explotation

Long Term Dependencies



Google DeepMind

RL problem

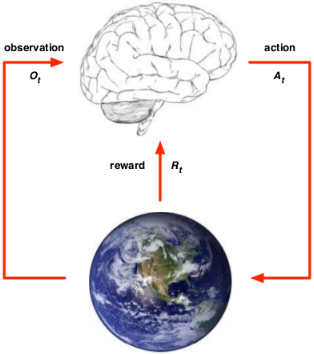
- **Reward** – scalar feedback signal
- **Goal** – pick the sequence of actions that maximizes the cumulative reward

Reward Hypothesis.

All goals can be described by the maximization of expected cumulative reward

Google DeepMind

- **Agent and Environment**



Source: `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf`

RL problem

- **History** – is sequence of observations and actions
- **State** – information used to decide what happens next (MDP/POMDP)

$$P(S_t|S_{t-1}) = P(S_t|S_1, ..S_{t-1})$$

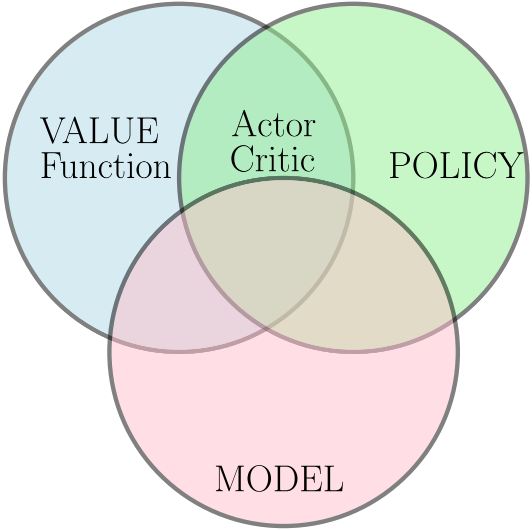
Inside the RL agents

An RL agent has one or more of these components:

- **Policy** – given a state provide a distribution over the actions
- **Value function** – given a state (state/action pair) estimate expected future reward
- **Model** – agent's representation of the world (planning)

Google DeepMind

RL agents taxonomy



Policy based methods (digression)

- Effective in high-dimensional / continous action spaces
- Can learn stochastic policies
- Better convergence properties
- Noisy gradients !

Reinforce

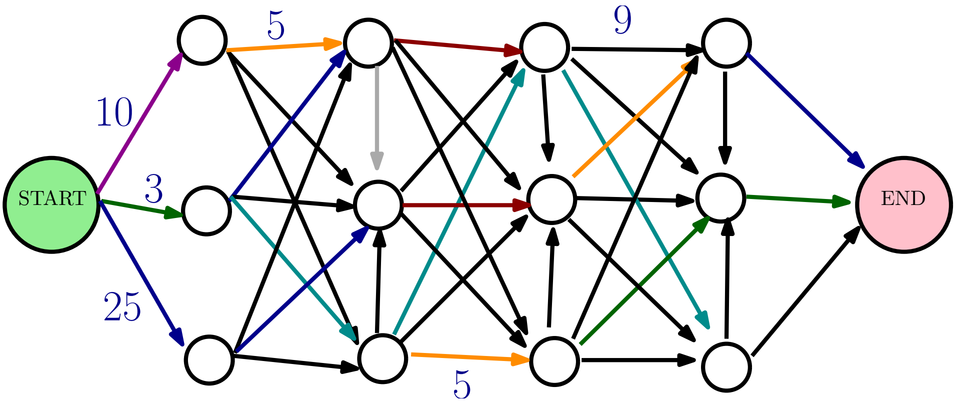
Directly maximize the cumulative reward !

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum d(s) \sum \pi_\theta(s, a) r_{s,a}$$

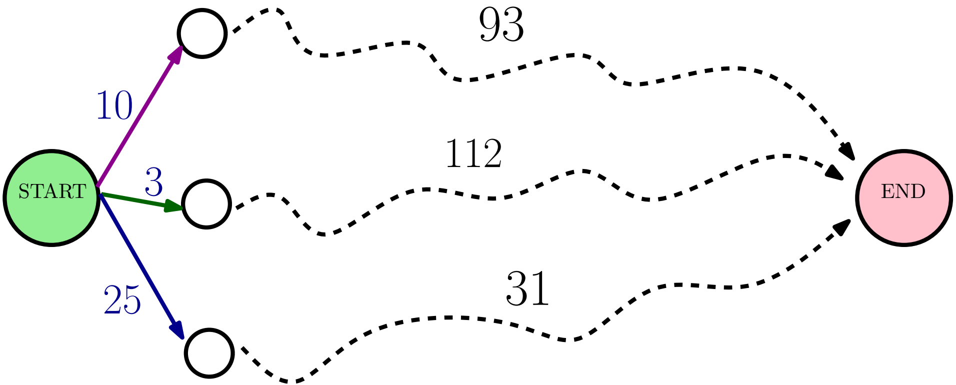
Maximize J. Using the log trick we have:

$$\frac{\partial J}{\partial \theta} = \sum d(s) \sum \pi_\theta(s, a) \frac{\partial \log \pi}{\partial \theta} r_{s,a} = \mathbb{E}_\pi[\frac{\partial \log \pi}{\partial \theta} r]$$

Primer Dynamic Programming



Primer Dynamic Programming



Bellman's Principle of Optimality
An optimal policy has the property that whatever initial state and initial decision are, the remaining decisions must consitute an optimal policy (Bellman, 1957)

$$V(x) = \max_{a \in \Gamma(x)} \{F(x, a) + \beta V(T(x, a))\}$$

Q-Values

The Q value $Q(x, a)$ is the expected cumulative reward for picking action a in state x.
We can act greedily or epsilon greedy.

$$\pi(a_i|x) = \begin{cases} 1 - \epsilon, & if Q(a_i, x) > Q(a_j, x) \forall j \\ \epsilon, & otherwise \end{cases}$$

Google DeepMind

Q-learning

Think of Q-values as the length of the path in the graph. Use dynamic programing (Bellman equation):

$$\hat{Q}_t(x_t, a_t) = r_{x_t, a_t} + \beta \max_a Q_t(x_{t+1}, a) \Rightarrow$$

$$\underbrace{Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \underbrace{\gamma}_{\text{learning rate}} \underbrace{(\hat{Q}(x_t, a_t) - Q_t(x_t, a_t))}_{\text{derivative the of square error}(\hat{Q}-Q)^2}}_{\text{Regress } Q \text{ to } \hat{Q} \text{ using SGD}}$$

Google DeepMind

Finally using deep learning for RL

What role does Deep Learning play in RL ?

- provides a compact form for Q (function approximator)

$$\theta_{t+1} = \theta_t + \gamma \underbrace{\left(\hat{Q}(x_t, a_t) - Q_{\theta_t}(x_t, a_t)\right)\frac{\partial Q_\theta}{\partial \theta}}_{\text{derivative of the square error } \frac{\partial(\hat{Q}-Q)^2}{\partial\theta}}$$

Google DeepMind

Q-learning in Theano? (theano pseudocode)

```
x = TT.vector("x")
q = TT.dot(Wout, TT.nnet.relu(TT.dot(W, x)+b)+bout)
forward = theano.function([x], q)
```

Q-learning in Theano? (theano pseudocode)

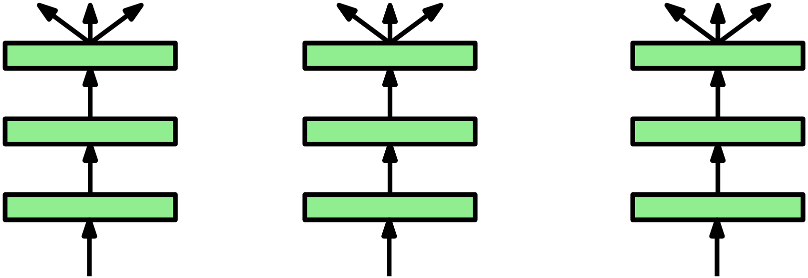
```
x = TT.vector("x")
q = TT.dot(Wout, TT.nnet.relu(TT.dot(W, x)+b)+bout)
params = [W,b,Wout, bout]
target_q = TT.scalar("target_q")
action = TT.iscalar('action')
lr = TT.scalar('lr')
gp = TT.grad((q[action] - target_q)**2, params)
learn = theano.function([x,action, target_q,lr], [],
        updates=[(p,p-lr*gp) for p,gp in zip(params,gp)])
```

Google DeepMind

Q-learning in Theano? (theano pseudocode)

```
for _,(x, x_tp1, act, reward) in enumerate(memory):
    target_q = reward + forward(x_tp1).max()
    learn(x, act, target_q, 1e-3)
```

But Learning can be tricky

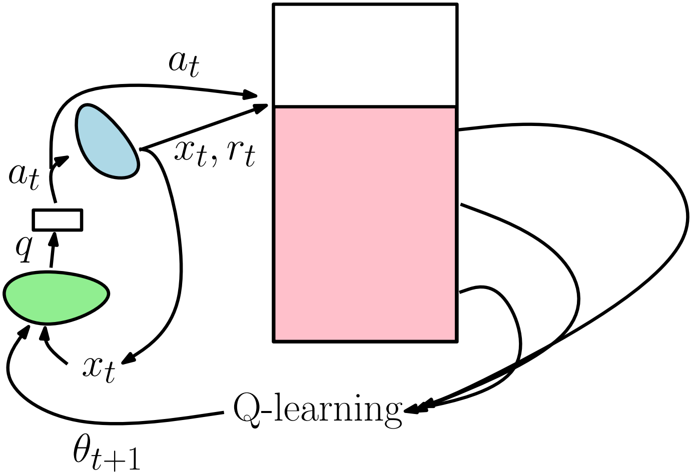


$$\mathbf{x}_t = 2 \qquad \mathbf{x}_{t+1} = 2 \qquad \mathbf{x}_{t+2} = 2$$

Correlated samples break learning

Google DeepMind

Solution: replay buffer



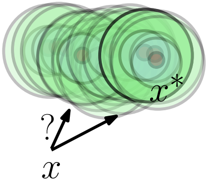
High variance and minibatches

- Reinforcement Learning is inherintly *sequential*
- Replay Buffer gives elegant solution to employ *minibatches*
- Minibatches means *reduced variance* in the gradients

Google DeepMind

Target network

- $\hat{Q}$ changes as fast as Q
- fix $\hat{Q}$ (target network) and update it periodically



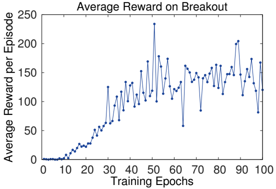
Moving target

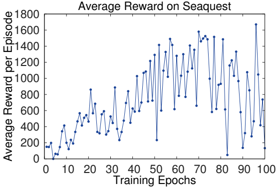
Other details

- SGD can be slow .. rely on RMSprop (or any new optimizer)
- Convolutional models are more efficient then MLPs
- DQN uses action repeat set to 4
- DQN receives 4 frames of the game at a time (grayscale)
- ϵ is anealled from 1 to .1
- Training takes time (roughly 12-14 days)

Google DeepMind

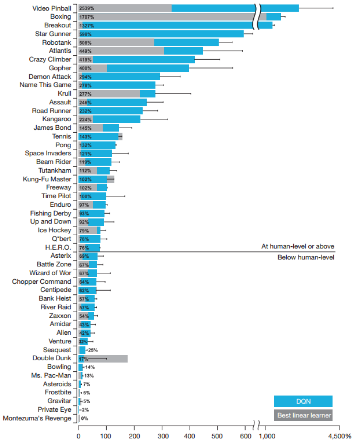
Results





Source: Mnih et al., Human-level control through deep reinforcement learning, Nature 2015

Results



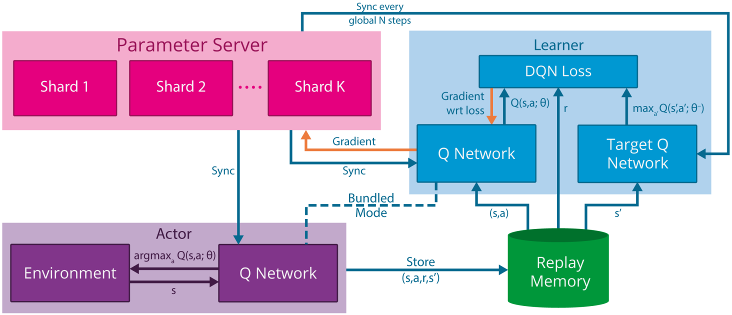
Source: Mnih et al., Human-level control through deep reinforcement learning, Nature 2015

Google DeepMind

Results

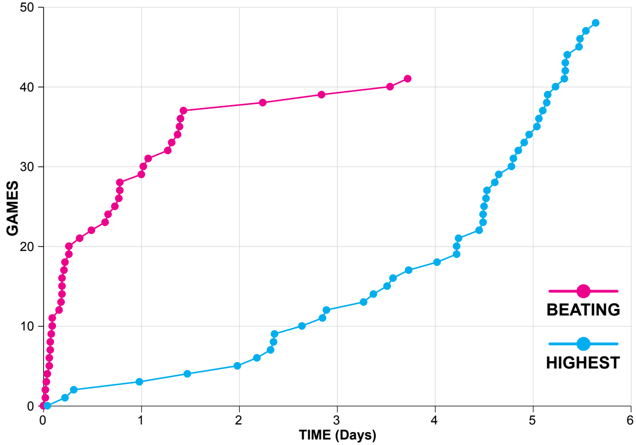
Nature paper videos

Parallelization – Gorila



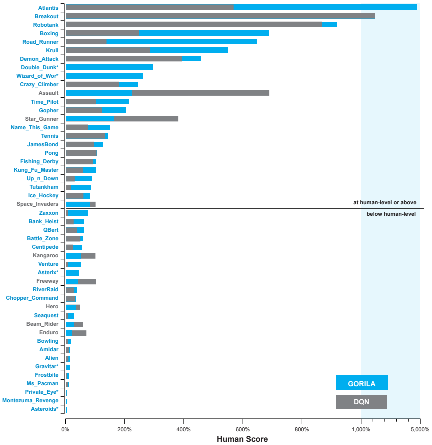
Source: Nair et al., Massively Parallel Methods for Deep Reinforcement Learning, ICML DL workshop

Results



Source: Nair et al., Massively Parallel Methods for Deep Reinforcement Learning, ICML DL workshop

Results



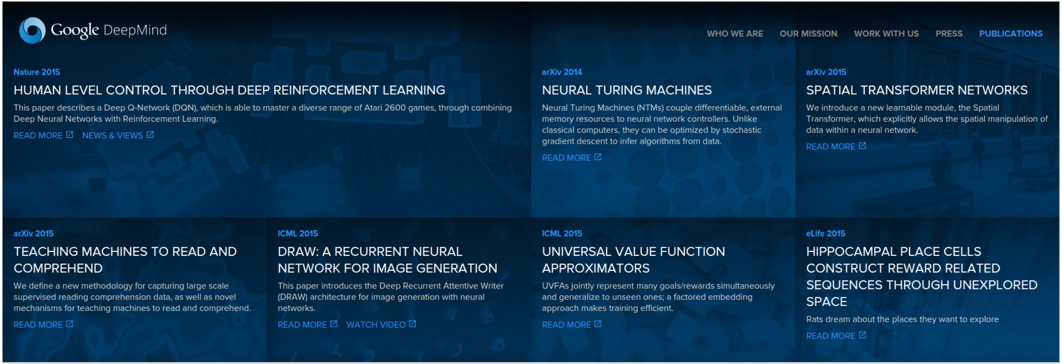
Source: Nair et al., Massively Parallel Methods for Deep Reinforcement Learning, ICML DL workshop

Where this doesn't work (straightforwardly)

- Continous control
- Robotics (experience is very explensive)
- Sparse rewards (Montezuma !?)
- Long term correlations (Montezuma !?)

But this does not mean that RL+DL can not be the solution !

DeepMind Research



Source: `http://deepmind.com/publications.html`

And we are hiring

Google DeepMind

WORK WITH US

WE ARE HIRING!

If you are an exceptional machine learning researcher, computational neuroscientist or software engineer, and want to be part of a world-class team working on the most exciting ground-breaking technology in an inspiring and collaborative environment then please get in touch.

`joinus@deepmind.com`

Google DeepMind

Thank you

Questions ?

Possible exercise for the afternoon sessions I

Pick one or several tasks from the Deep Learning Tutorials:

- Logistic Regression
- MLP
- AutoEncoders / Denoising AutoEncoders
- Stacked Denoising AutoEncoders

Google DeepMind

Possible exercise for the afternoon sessions II

Compare different initialization for neural networks (MLPs and ConvNets) with rectifieres or tanh. In particular compare:

- The initialization proposed by Glorot et al.
- Sampling uniformally from $[-\frac{1}{fan_{in}}, \frac{1}{fan_{in}}]$
- Setting all singular values to 1 (and biases to 0)

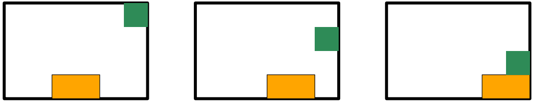
How do different optimization algorithms help with this initializations? Extra kudos for interesting plots or analysis. Please make use of the Deep Learning Tutorials.

Possible exercise for the afternoon sessions III
Requires convolutions

Re-implement the AutoEncoder tutorial using convolutions both in the encoder and the decoder. Extra kudos for allowing pooling (or strides) in the encoder.

Possible exercise for the afternoon sessions IV
Requires Reinforcement Learning

Attempt to solve the Catch game.



Not actual screenshots of the game