DTU Course 02456 Deep learning
6 Reinforcement learning

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

September 27, 2016

Objectives of lecture 6

- P1: Reinforcement learning
- What it is and can be used for
- P2: An introductory example - AlphaGo
- P3: Policy gradient methods
- P4: A few final words

Part 1:
Reinforcement learning

Reinforcement learning

- An agent learning to take actions in an environment so as to maximize some notion of cumulative reward.
- Actions taken now will affect future reward.

Reinforcement vocabolary

- s = state = environment state, $s \in S$
- a = action, $a \in A$

Reinforcement vocabolary

- s = state = environment state, $s \in S$
- a = action, $a \in A$
- Transition: $s_{t+1} = \mathrm{Env}(s_t, a_t)$
- Reward: $r_{t+1} = \mathrm{Reward}(s_t, a_t, s_{t+1})$
- Observation: $o_t = (r_t, s_t)$

Reinforcement vocabolary

- s = state = environment state, $s \in S$
- a = action, $a \in A$
- Transition: $s_{t+1} = \mathrm{Env}(s_t, a_t)$
- Reward: $r_{t+1} = \mathrm{Reward}(s_t, a_t, s_{t+1})$
- Observation: $o_t = (r_t, s_t)$
- π = Policy = distribution over a given a history

Reinforcement vocabolary

- s = state = environment state, $s \in S$
- a = action, $a \in A$
- Transition: $s_{t+1} = \mathrm{Env}(s_t, a_t)$
- Reward: $r_{t+1} = \mathrm{Reward}(s_t, a_t, s_{t+1})$
- Observation: $o_t = (r_t, s_t)$
- π = Policy = distribution over a given a history
- Cumulative (discounted) reward/return

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

- Expected cumulative reward

$$\rho^{\pi} = \mathbb{E}[R|\pi] \ .$$

Reinforcement learning approaches (thanks Wikipedia!)

- Brute force - go through all possible policies π and estimate ρ^{π} by sampling.

Reinforcement learning approaches (thanks Wikipedia!)

- Brute force - go through all possible policies π and estimate ρ^π by sampling.
- Value function approaches
  - Value function $V^\pi(s) = \mathbb{E}[R|s, \pi]$
  - Q-function

$$Q^\pi(s, a) = \mathbb{E}[R|s, a, \pi]$$

Reinforcement learning approaches (thanks Wikipedia!)

- Brute force - go through all possible policies π and estimate ρ^π by sampling.
- Value function approaches
  - Value function $V^\pi(s) = \mathbb{E}[R|s, \pi]$
  - Q-function
$$Q^\pi(s, a) = \mathbb{E}[R|s, a, \pi]$$
- Monte Carlo methods
  - Policy estimation - estimate $Q^\pi(s, a)$
  - Policy improvement - update π based upon estimate

Reinforcement learning approaches (thanks Wikipedia!)

- Brute force - go through all possible policies π and estimate ρ^π by sampling.
- Value function approaches
  - Value function $V^\pi(s) = \mathbb{E}[R|s, \pi]$
  - Q-function
$$Q^\pi(s, a) = \mathbb{E}[R|s, a, \pi]$$
- Monte Carlo methods
  - Policy estimation - estimate $Q^\pi(s, a)$
  - Policy improvement - update π based upon estimate
- Temporal difference methods - change policy on the fly.

Reinforcement learning approaches (thanks Wikipedia!)

- Brute force - go through all possible policies π and estimate ρ^{π} by sampling.
- Value function approaches
  - Value function $V^{\pi}(s) = \mathbb{E}[R|s, \pi]$
  - Q-function
    $$Q^{\pi}(s, a) = \mathbb{E}[R|s, a, \pi]$$
- Monte Carlo methods
  - Policy estimation - estimate $Q^{\pi}(s, a)$
  - Policy improvement - update π based upon estimate
- Temporal difference methods - change policy on the fly.
- Direct policy search
  - Make parametrised model for policy: π_{θ}
  - Optimise (estimate of)
    $$\rho^{\pi_{\theta}} = \mathbb{E}[R|\pi_{\theta}]$$
    wrt to θ
  - Gradient based: REINFORCE
  - Not gradient based (e.g. evolutionary): NEAT

Part 2:
Deepmind's AlphaGo

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

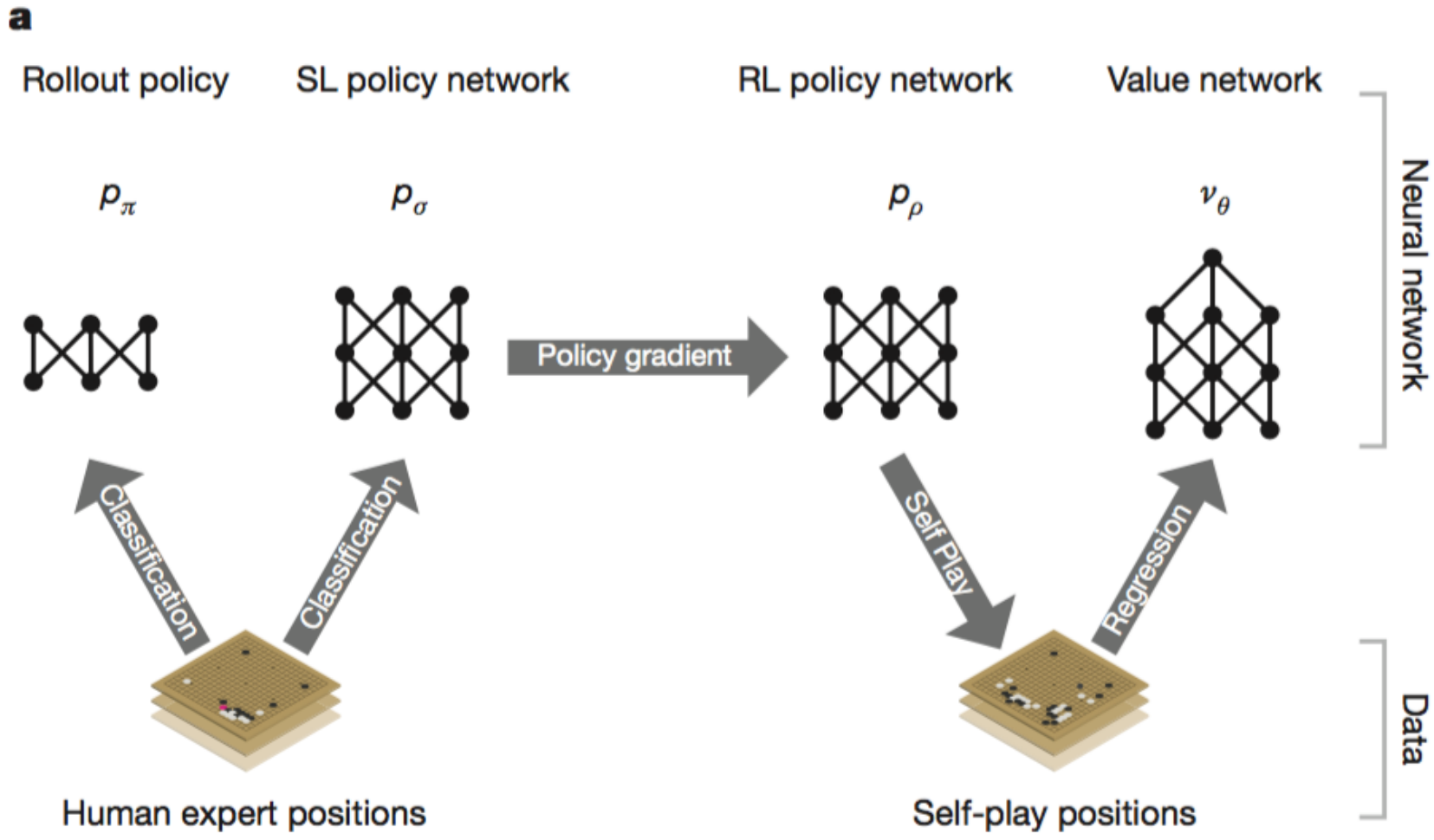
David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

Value function $v^*(s)$ = outcome for optimal play

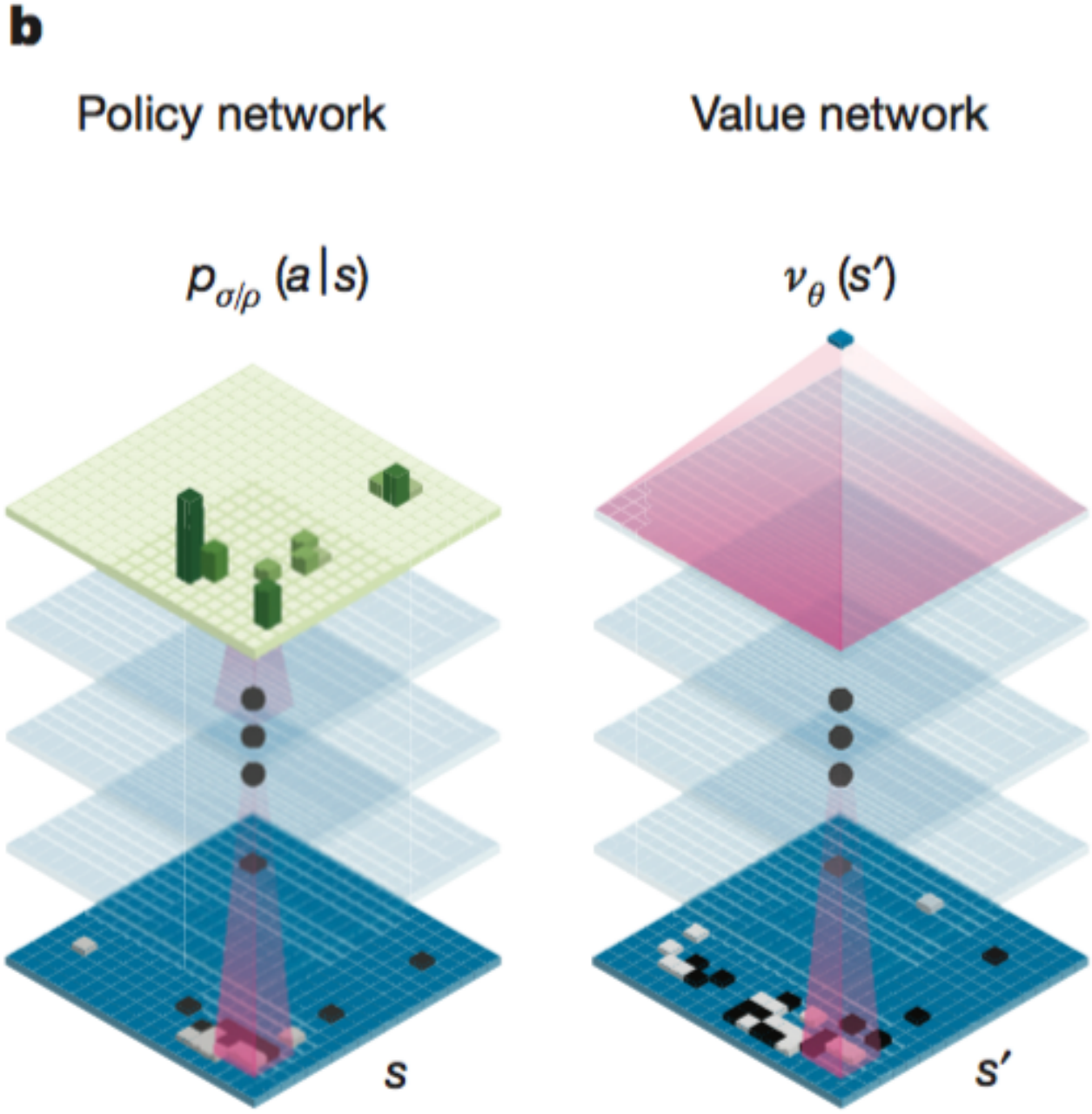
Value function $v^*(s)$ = outcome for optimal play

- s = state = game position
- Game breadth: b
- Game depth. d
- Complexity: b^d for calculating $v^*(s)$
- Chess: $b \approx 35$, $d \approx 80$
- Go: $b \approx 250$, $d \approx 150$

Rollout policy, SL and RL policy and value networks



Policy and value networks



Step 1: supervised learning (SL) policy $p_\sigma(a|s)$

- a = action
- s = state
- We have large database of expert games.
- Train classifier to imitate expert moves (s, a):

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial\sigma}$$

- Supervised learning (SL) policy

Step 2: reinforcement learning (RL) policy $p_\rho(a|s)$

- The policy network $p_\rho(a|s)$ plays against (a younger version of itself).
- Record whether it wins/losses: $z_t = r(T) = +1/-1$
- Train classifier to imitate expert moves (s, a):

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t$$

- Better than SL policy.

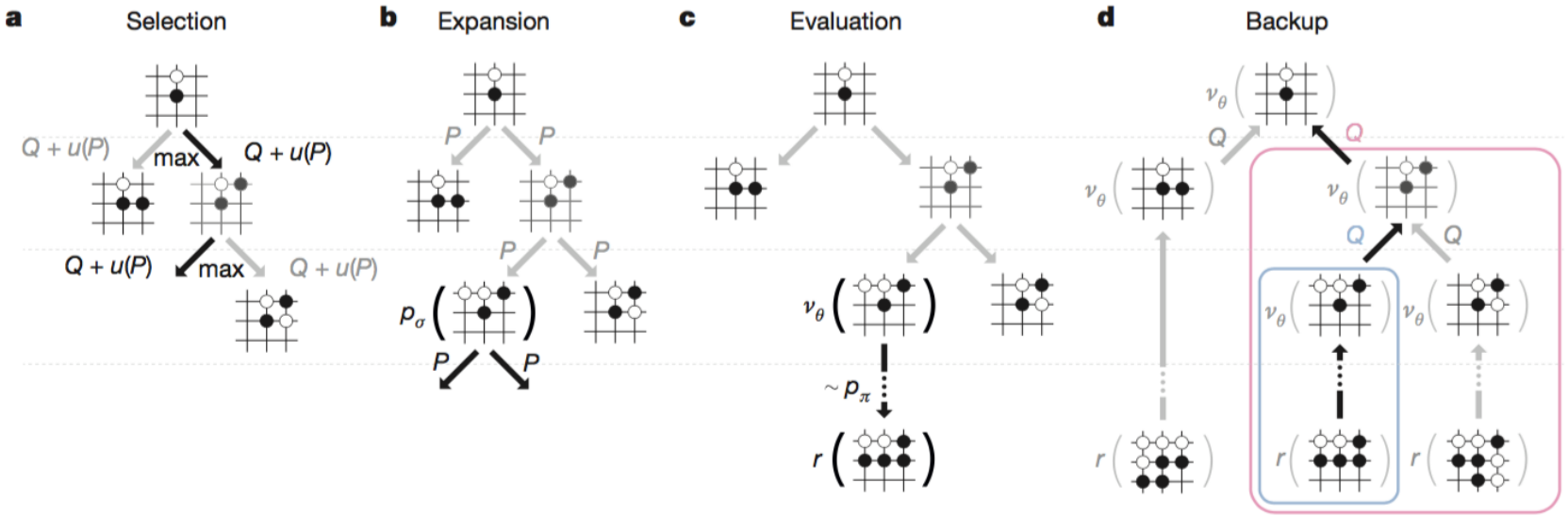
Step 3: RL value function $\nu_\theta(s)$

- Remember that value function $v^*(s)$ is unknown.
- We can try to learn the value function of our RL policy network by a network: $\nu_\theta(s)$:

$$\Delta\theta \propto \frac{\partial \log \nu_\theta(s)}{\partial\theta}(z - \nu_\theta(s))$$

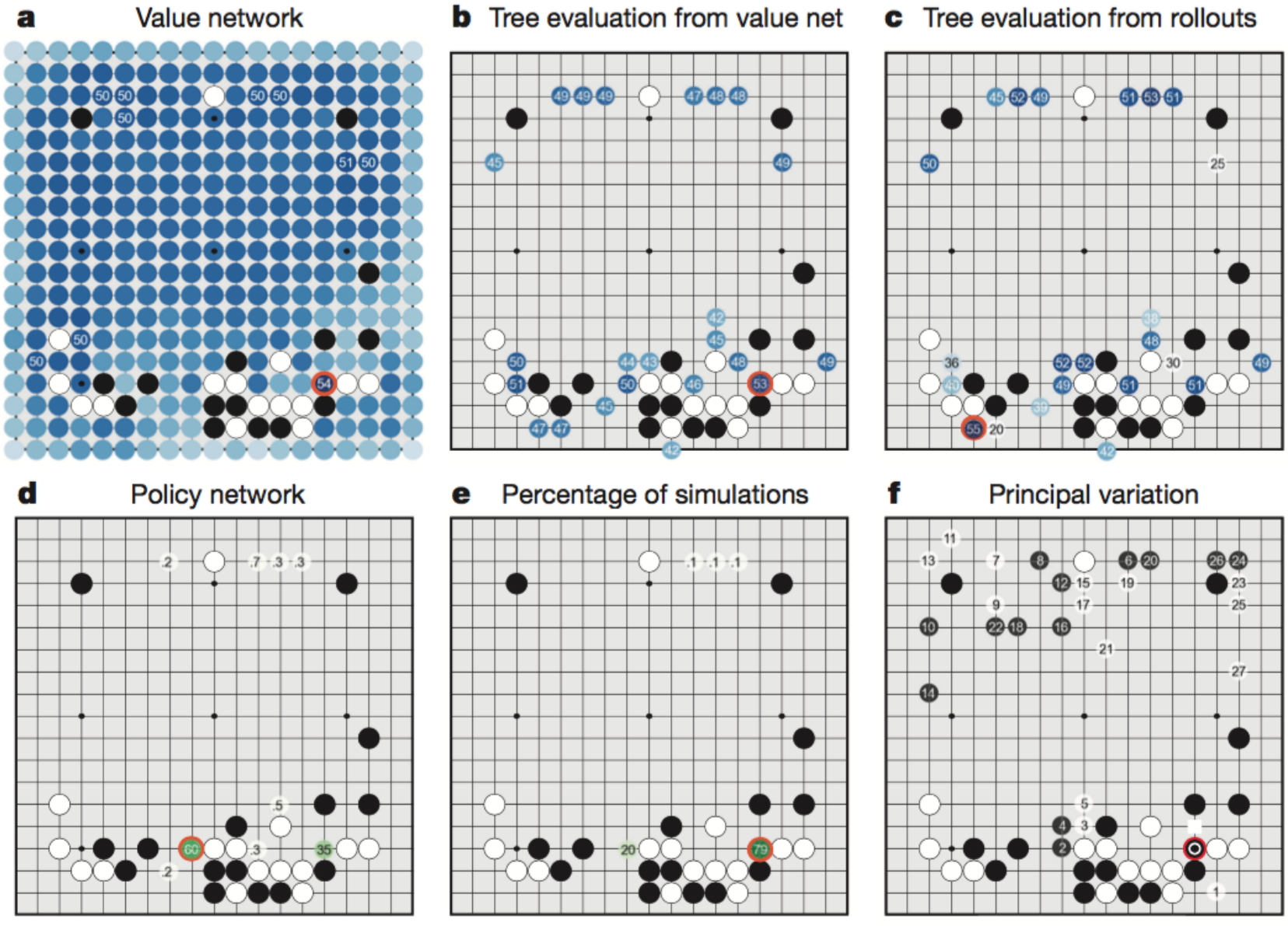
- We can use this as an ingredient in a Monte Carlo tree search
- to score positions s

Step 4: Monte Carlo tree search



- SL (and not RL) policy network used for to propose moves in tree search
- SL better for exploration!
- RL value function used for scoring positions.

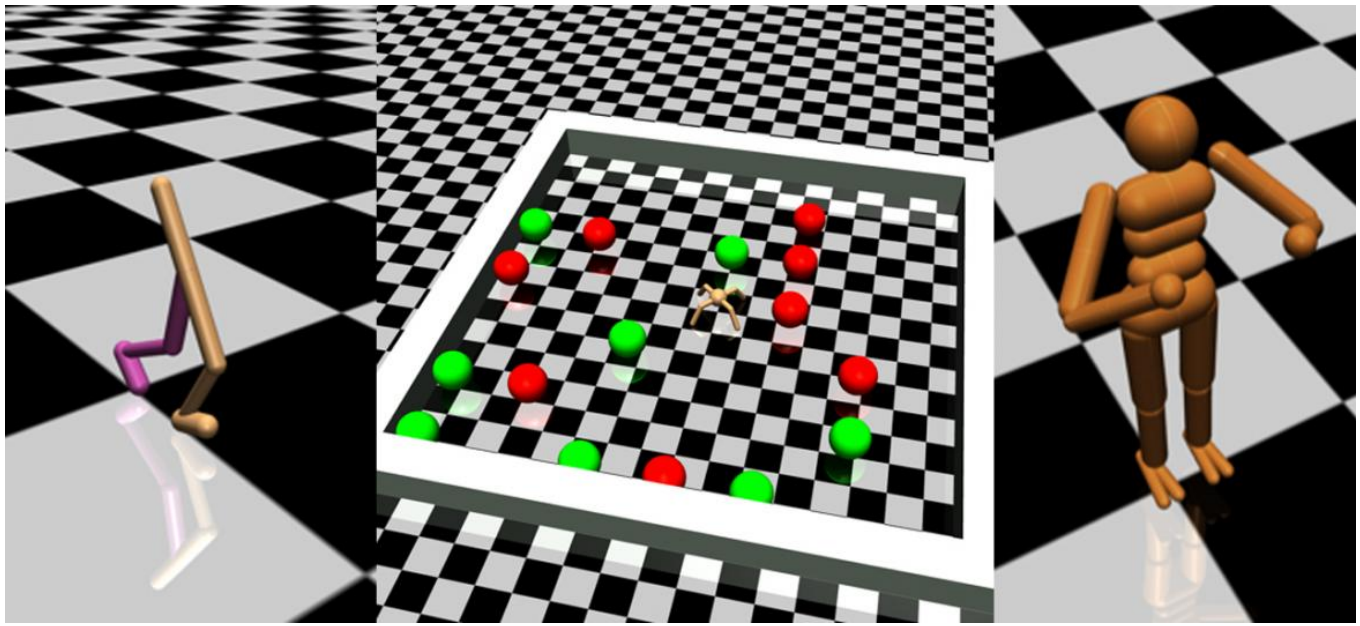
An example


a Value network
b Tree evaluation from value net
c Tree evaluation from rollouts
d Policy network
e Percentage of simulations
f Principal variation

Part 3:
Policy gradients

Policy gradients

- Policy gradients is arguably the simplest RL approach
- RL policy network in AlphaGo is an example
- Use $r_t = 0$ $t < T$ and $r_T = z = \pm 1$.



- Let us formalise this a bit on the next slide
- using the notation from the AlphaGo

Policy gradients

- We have a policy network $p_\theta(a|s)$
- We sample roll-outs:

$$\mathbf{a} = a_1, \ldots, a_T \sim p_\theta(\mathbf{a}|\mathbf{s}) = \prod_{t=1}^{T} p_\theta(a_t|s_{t-1}) \ .$$

- Our expected discounted cumulative reward is

$$\mathbb{E}[R|\theta] = \int_{\text{roll}-\text{outs}} R(\mathbf{a}) p_\theta(\mathbf{a}|\mathbf{s}) d\mathbf{a}$$

Policy gradients

- We have a policy network $p_\theta(a|s)$
- We sample roll-outs:

$$\mathbf{a} = a_1, \ldots, a_T \sim p_\theta(\mathbf{a}|\mathbf{s}) = \prod_{t=1}^{T} p_\theta(a_t|s_{t-1}) \ .$$

- Our expected discounted cumulative reward is

$$\mathbb{E}[R|\theta] = \int_{\text{roll-outs}} R(\mathbf{a}) p_\theta(\mathbf{a}|\mathbf{s}) d\mathbf{a}$$

- Take gradient wrt θ and use

$$\nabla_\theta p_\theta(\mathbf{a}|\mathbf{s}) = p_\theta(\mathbf{a}|\mathbf{s}) \nabla_\theta \log p_\theta(\mathbf{a}|\mathbf{s})$$

  and replace integral by and average over S roll-outs

$$\nabla_\theta \mathbb{E}[R|\theta] \approx \frac{1}{S} \sum_{s=1}^{S} R(\mathbf{a}^{(s)}) \nabla_\theta \log p_\theta(\mathbf{a}^{(s)}|\mathbf{s}^{(s)})$$

- The small print: Gradient estimator has high variance
- but we can correct for that!

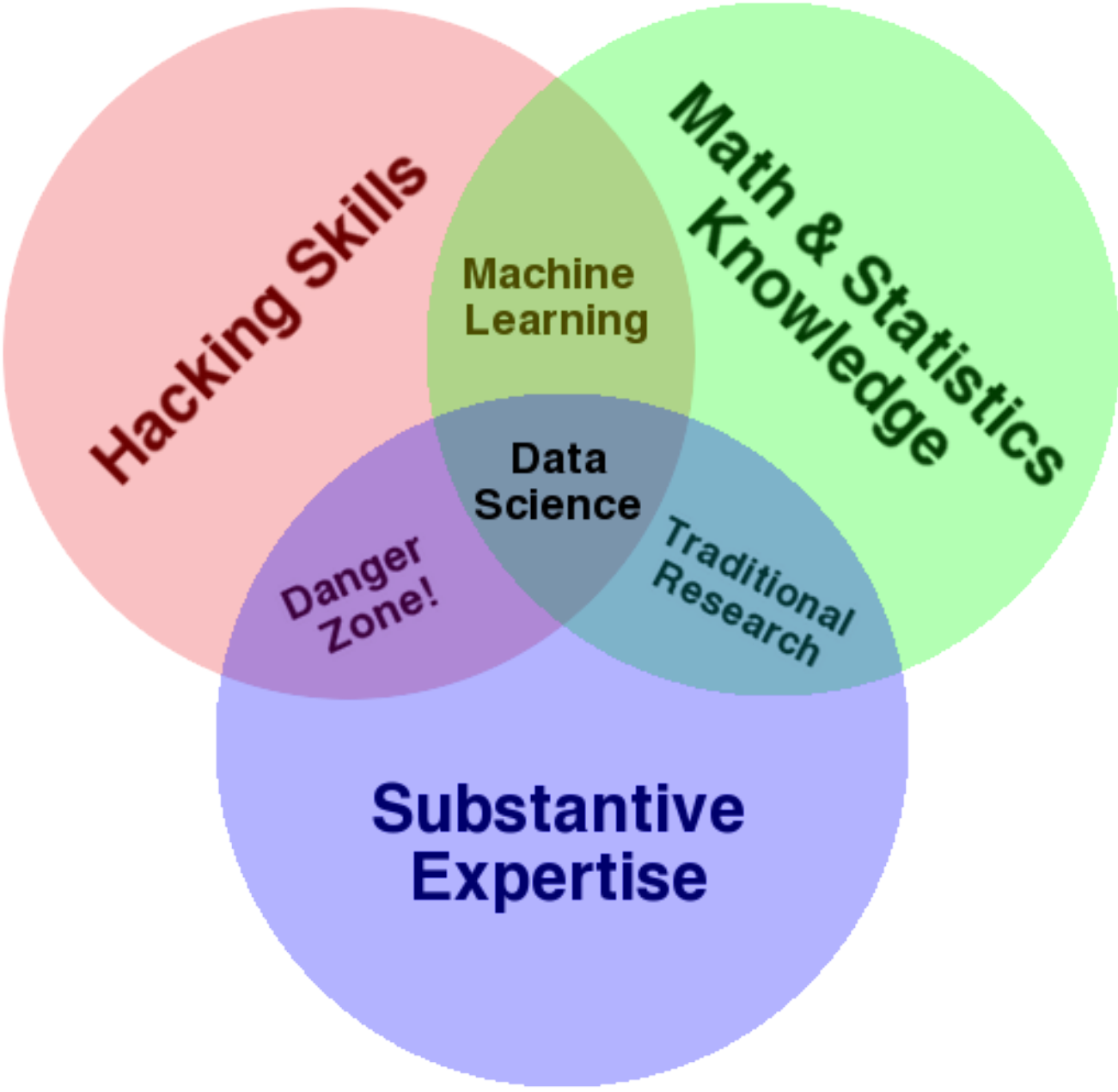
Part 4:
A few final words

Does it scale?

Joaquin Quiñonero Candela, Director of Applied Machine Learning at Facebook on Quora:

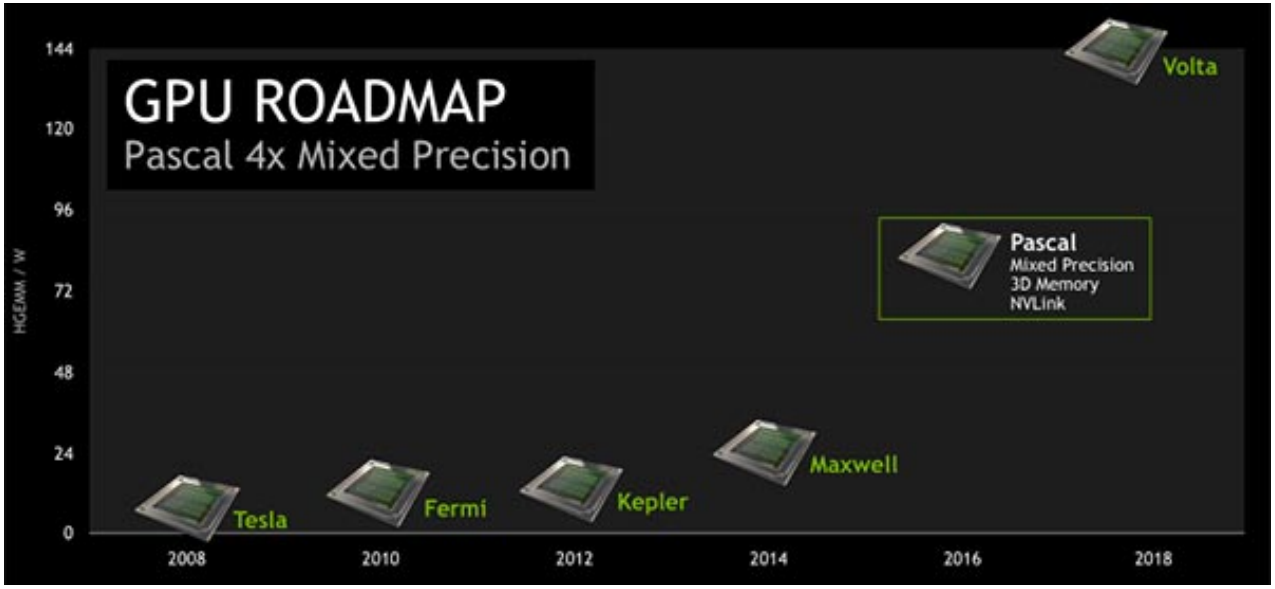
- In computer vision we have a system that processes every single image and video. . . well over 1B items per day.
- . . . predict the content. . . . . to generate captions for the blind,. . . detect and take down offensive content, improve media search results, automate visual captcha. . . .
- In language technology,. . . translate over 2B posts every single day, with over 1800 language directions representing more than 40 unique languages.
- . . . these models are used to rank news feed stories (1B users every day, 1.5K stories per user per day on average), ads, search results (1B+ queries a day), trending news, friend recommendations and even rank notifications that a user receives, or rank the comments on a post.

Data science - we are getting there



Deep learning prediction about the future

- Expect rapid progress in coming years!
- As you know by now it is still hard to make it work!



Thanks!
Ole Winther