Martin Mundt, Dr. Iuliia Pliushch, Prof. Dr. Visvanathan Ramesh

Pattern Analysis & Machine Intelligence Praktikum: MLPR-WS19

Week 14: Metalearning with Reinforce

On-policy vs. off-policy RL algorithms

On-policy	*Off-policy*
Agent can pick actions.	Agent can’t pick actions.
Agent always follows his **own** policy.	Learning with exploration, playing without. For example, learning from experts or recorded sessions.
Example: reinforce (policy gradient method)	Example: Q-learning (value-based method)

https://www.coursera.org/learn/practical-rl/home/welcome

Policy

$$\pi - policy$$
$$s - state$$
$$a - action$$

Deterministic $a = \pi_\theta(a\|s)$	Stochastic $a \sim \pi_\theta(a\|s)$

Which kind of policy would you choose if you wanted to

Learn to win rock-paper-scissors?

https://www.coursera.org/learn/practical-rl/home/welcome

Policy gradient methods: objective

- Maximize expected reward **J**
- **J** is the expectation over states and actions of earned reward **R**
- **p(s)** – probability of visiting a certain state
- **π_θ(a|s)** – policy stochasticity: probability of choosing an action **a** in state **s**

$$J = E_{s\sim p(s),\, a\sim\pi_\theta(a|s)} R(s,a)$$

$$= \int_s p(s) \int_a \pi_\theta(a|s) R(s,a)\, da\, ds$$

State visiting frequency
(policy-dependent)

Reward for 1-step session
Assumption: the session takes only 1 step

https://www.coursera.org/learn/practical-rl/home/welcome

Policy gradient methods: derivative

- What is the analytical **derivative of J** which can be easily approximated with **sampling**?

$$J = E_{s\sim p(s),\, a\sim \pi_\theta(a|s)} R(s,a)$$

$$= \int_s p(s) \int_a \pi_\theta(a|s) R(s,a)\, da\, ds$$

$$\nabla \pi(z) = \pi \nabla \log \pi(z)$$

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(s|a) R(s,a)\, da\, ds$$

https://www.coursera.org/learn/practical-rl/home/welcome

Policy gradient methods: reward

- How should we change **J** if our session/game takes more than one step to complete?

$$J = E_{s \sim p(s), a \sim \pi_\theta(a|s)} \log \pi_\theta(a|s) R(s,a) \, da \, ds$$

θ_0

- substitute the **reward R** with **cumulative discounted reward G**

$$G_t = \sum_{i=0}^{\infty} \gamma^i * r_{t+i}$$

https://www.coursera.org/learn/practical-rl/home/welcome

Policy gradient methods: Reinforce

- initialize NN weights θ_0
- Loop:
  - Sample N sessions z_i under current $\pi_\theta(a|s)$
  - Evaluate policy gradient

Cumulative discounted reward

θ_0

$$J = E_{s\sim p(s), a\sim\pi_\theta(a|s)} \log \pi_\theta(a|s) G(s,a)\, da\, ds =$$

$$= \frac{1}{M} \sum_{z_i \in Z} \sum_{s,a \in z_i} \log \pi_\theta(a|s) G(s,a)$$

- Ascend

$$\theta_{i+1} \leftarrow \theta_i + \alpha \nabla J$$

https://www.coursera.org/learn/practical-rl/home/welcome

RL application: Neural architecture search

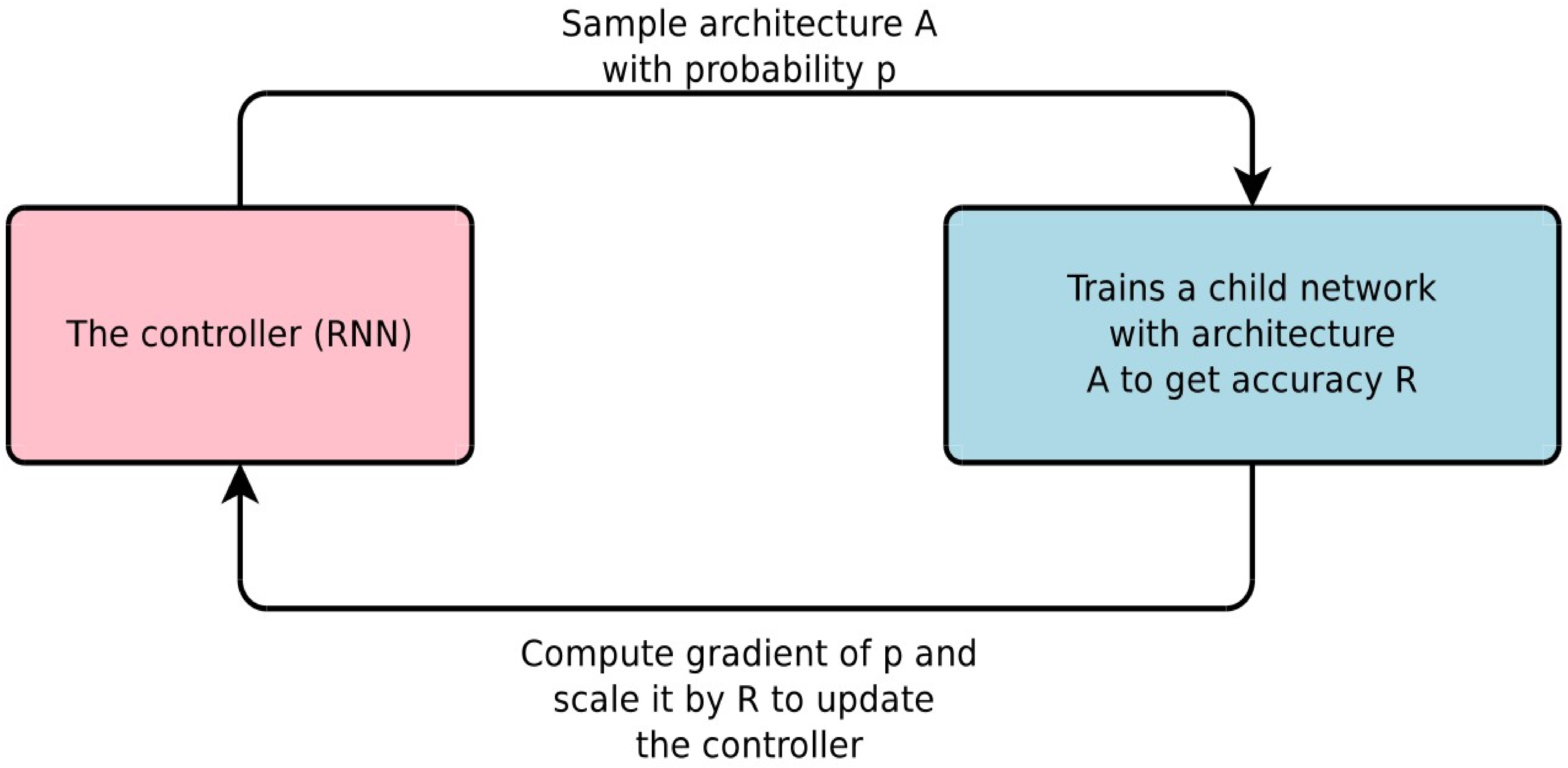
- Suppose we wanted to find out what is the best **neural architecture** to solve a problem, for example MNIST digit recognition

- How would our **environment** look? What are the **states**, **actions** and **rewards**?

- What are the neural architecture **parameters** that we could change?

States: encode parameter choices
Actions: fix parameter values
Rewards: accuracy of the network

Meta-Learning:
Neural architecture search



Sample architecture A
with probability p

The controller (RNN)

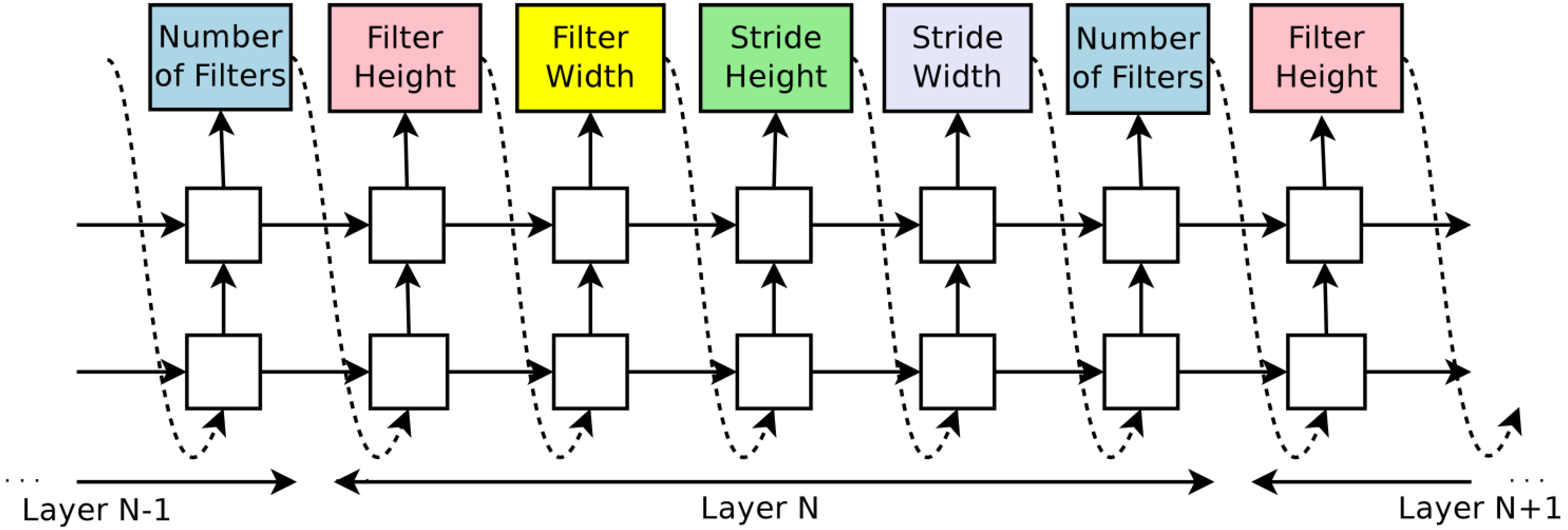
Trains a child network
with architecture
A to get accuracy R

Compute gradient of p and
scale it by R to update
the controller

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." ICLR (2017).

Meta-Learning:
Neural architecture search

- Suppose we use an **LSTM** cell, what is the sequence it should learn?
- The **sequence** starts with predicting the parameters of the **first layer** and ends with the **last**, such that given the sequence, a **NN** can be constructed, trained on a chosen task and the **accuracy** computed.



Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." ICLR (2017).

Meta-Learning:
Neural architecture search

- Suppose we use an **LSTM** cell, what is the sequence it should learn?

- The **sequence** starts with predicting the parameters of the **first layer** and ends with the **last**, such that given this sequence, a **NN** can be constructed, trained on a chosen task and the **accuracy** computed.

- Hence, how often do we get **rewards** from the environment, if a step in it involves fixing **one** of the neural architecture parameters?

►The rewards we get are **sparse**.