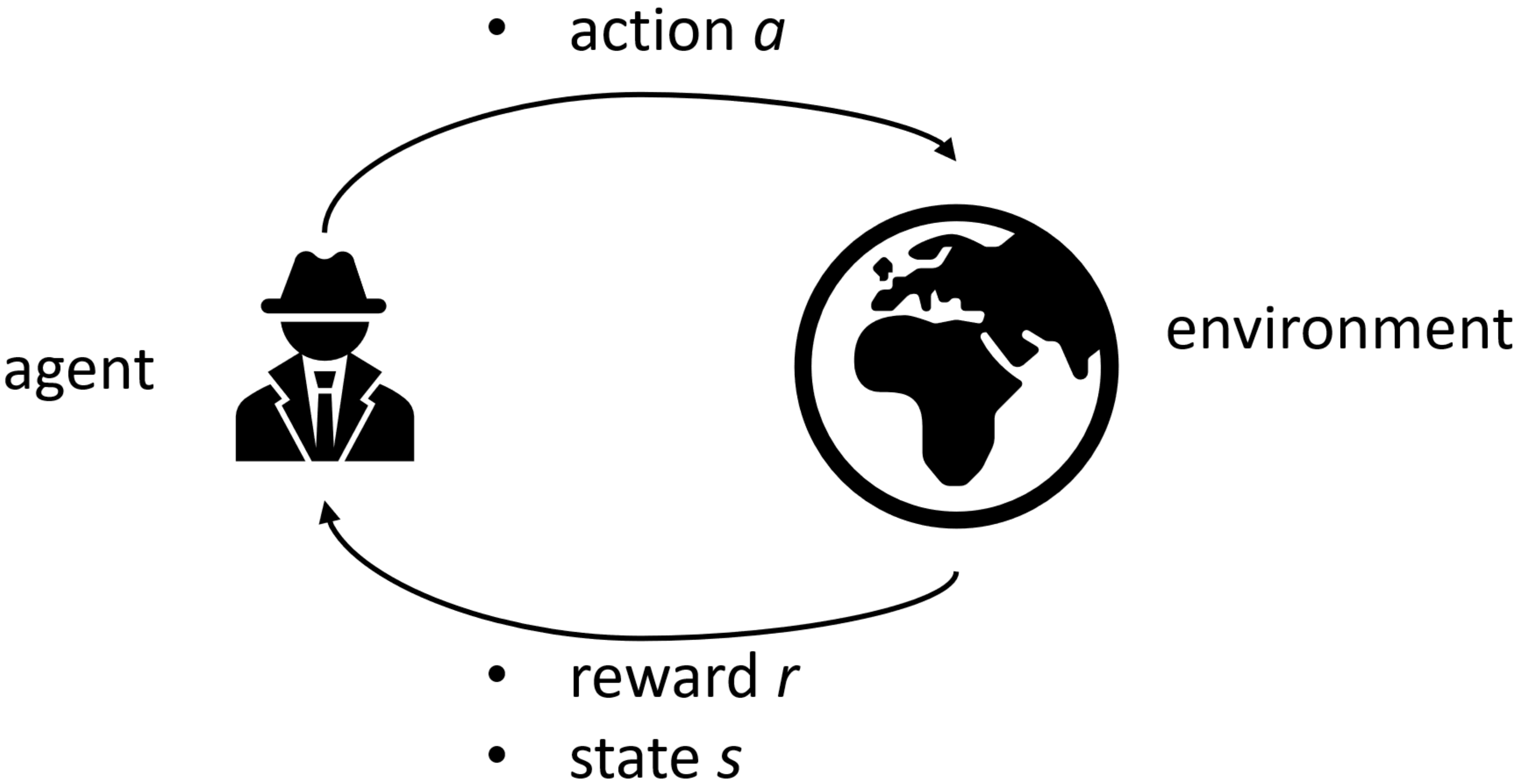
Tuning a reinforcement learning algorithm

Martin Zaefferer

July 20, 2021

Reinforcement learning …

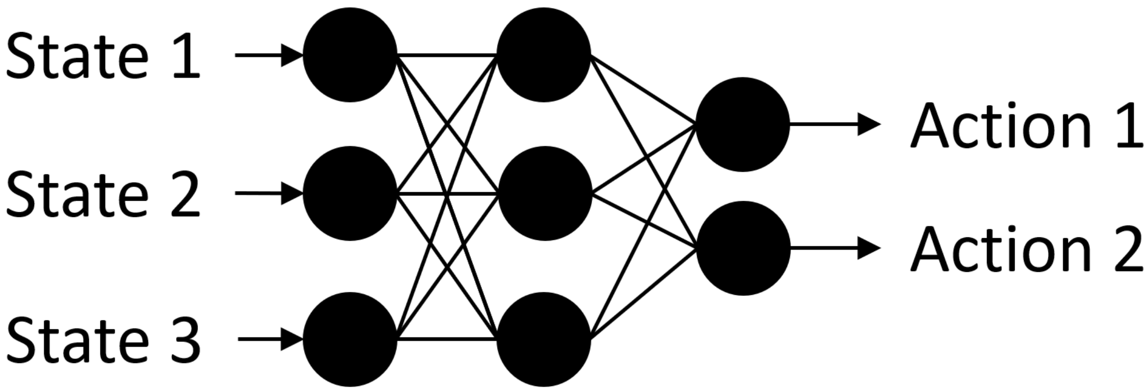
Reinforcement learning



… with Neural Networks

Neural Networks in reinforcement learning

- ‘Policy’: map states to actions
- Can have various shapes and forms
  - Tables, decision trees, …
- Of special interest here:
  - Artificial neural networks (NN)
  - Weights of the NN are adapted / updated during learning



Influencing performance

- Parameters of the learning algorithm
  - E.g., discount factors, step sizes, initial values, …
- Parameters of the NN
  - E.g., number of units, number of layers, learning rates, …
- Other parameters
  - E.g., agent morphology, environmental parameters
- Question is:
  - How to set the parameters correctly?
- Answer:
  - Tuning / optimization

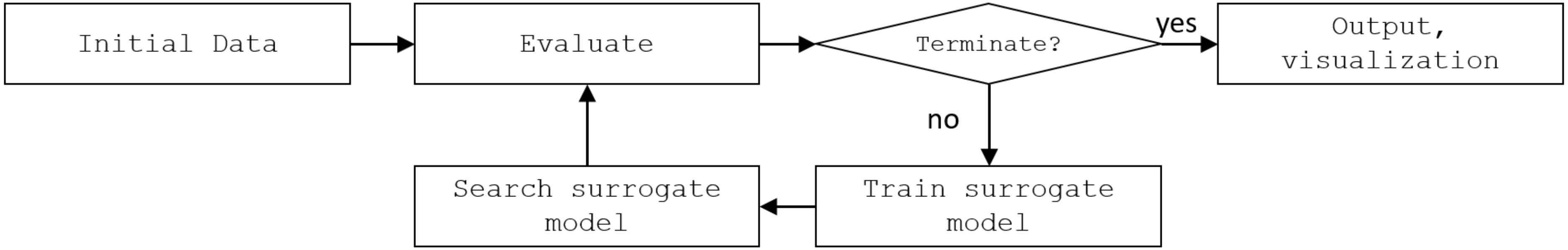
Limitations!

Evaluation costs

- NN training can be costly
  - Thousands or millions of weights to set
- Evaluating the agent in the environment may also be costly
  - E.g., observing a robot trying to solve a complex task
- Costs can easily become prohibitive for tuning
- Solution: surrogate model-based optimization

Surrogate model-based optimization

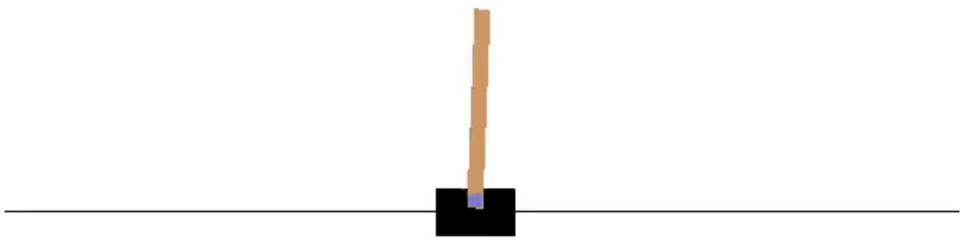
- Surrogate model
  - Learns relation between parameters and performance
- Evaluate the surrogate model instead of the actual problem



Demonstrative experiment

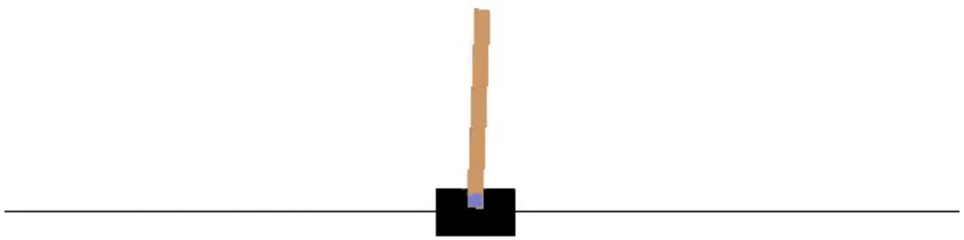
RL Environment: CartPole

- Cart balances pole
  - openai gym: CartPole-v0
- States: 4, continuous
  - Cart position & velocity
  - Pole angle & velocity
- Actions: 2, discrete
  - Push cart left
  - Push cart right



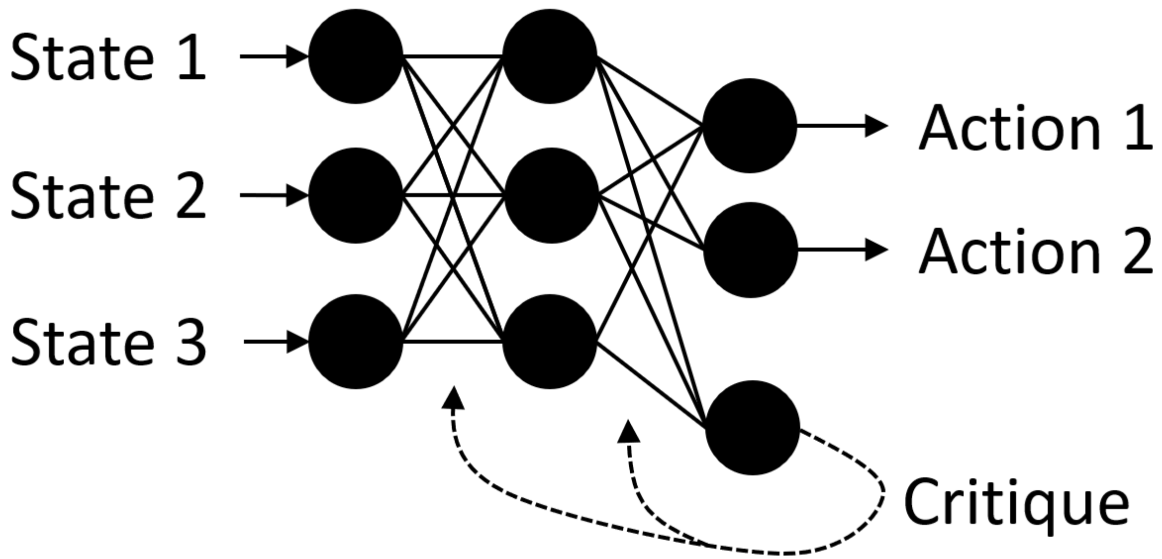
RL Environment: CartPole

- Reward:
  - 1 for each time step
- Stop if:
  - Pole angle outside +/- 12 degree
  - Cart position outside +/- 2.4
  - After 200 time steps
- Learning problem solved if:
  - Average reward of 195.



RL Agent: Actor-Critic

- Actor maps states to actions
- Critic learns 'value' of states (via rewards)
- Improve actor to beat critic estimation



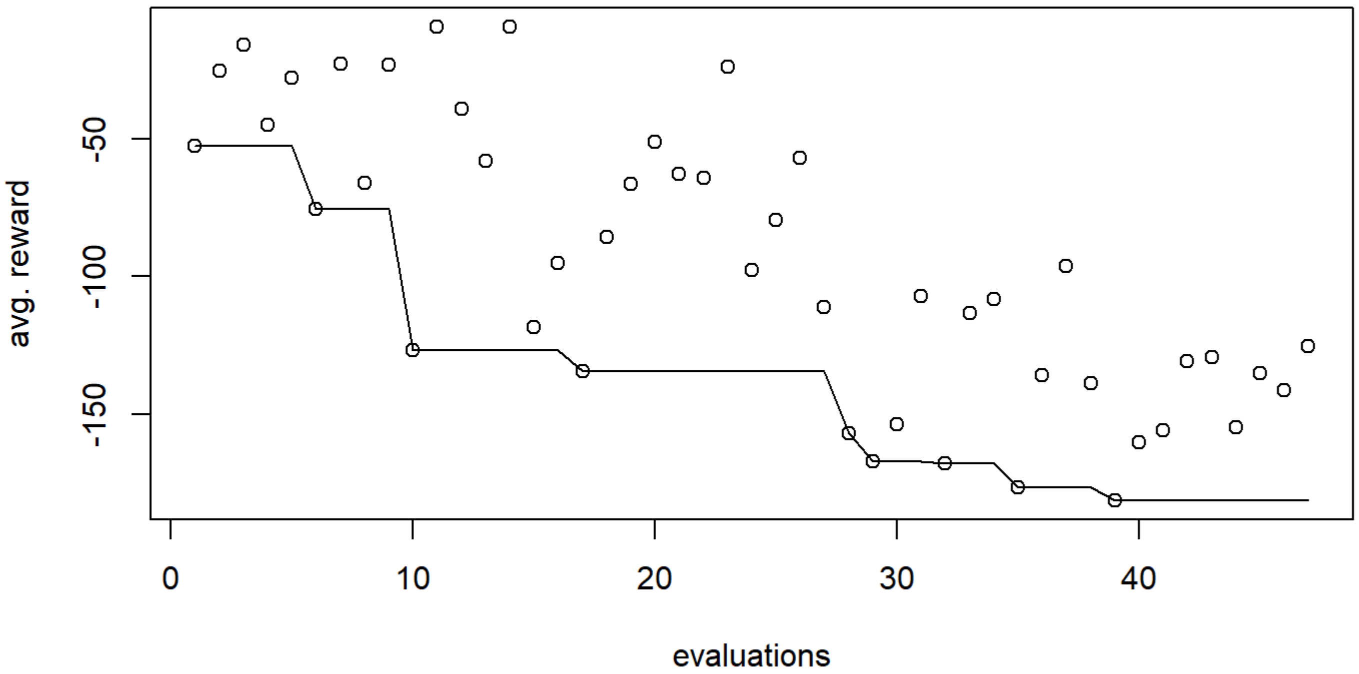
Example by Apoorv Nandan, 2020/05/13, https://keras.io/examples/rl/actor_critic_cartpole/

RL Tuning

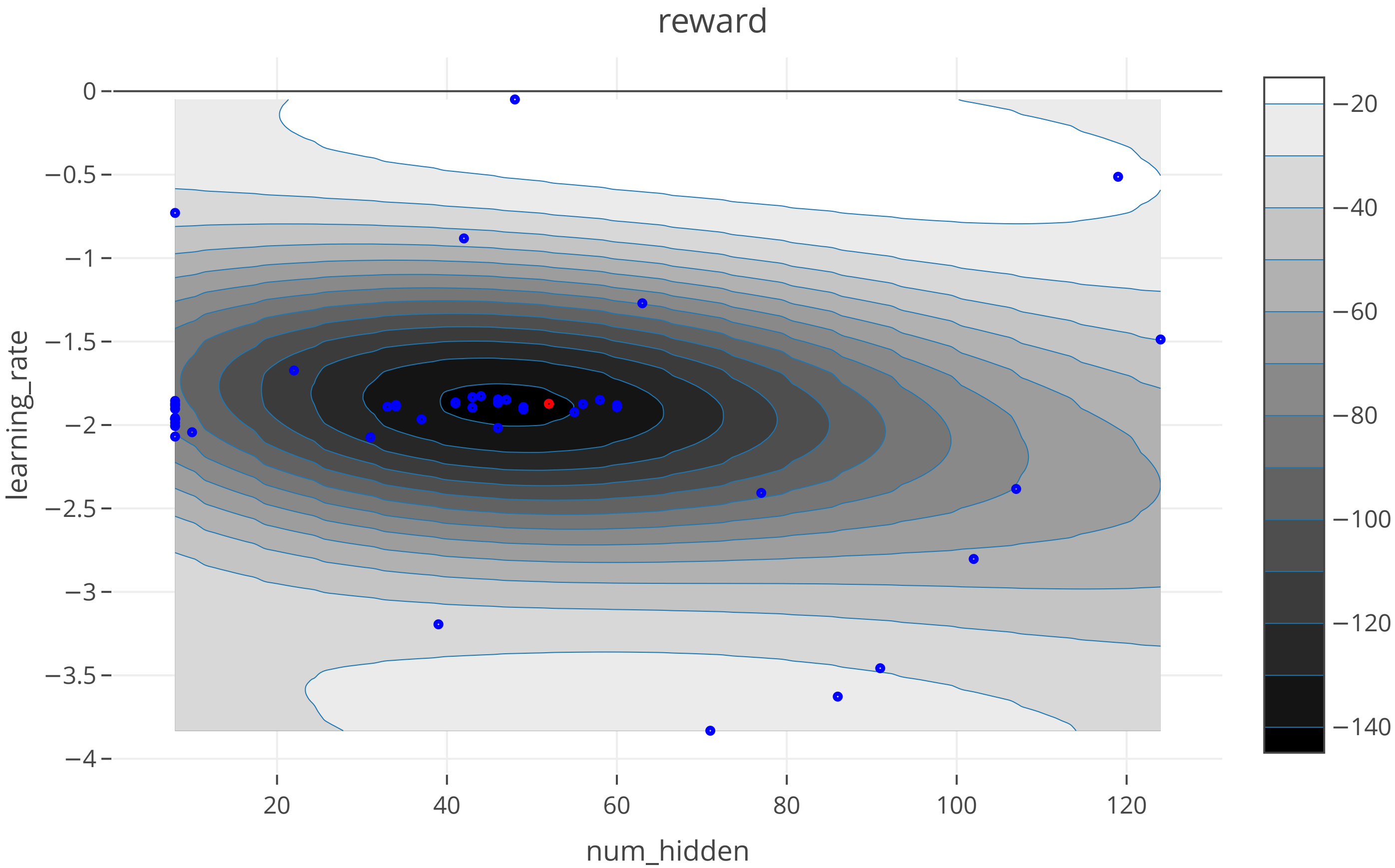
- Tuned Parameters
  - num_hidden NN units: 8, …, 128
  - learning_rate of NN (log_{10} scale): -4, …, 0
  - gamma: Actor-Critic discount factor: 0.5, …, 1.0
- Surrogate model: Gaussian process
- 2 hours runtime budget
- 5 replications for each parameter configuration

Results: optimization progress

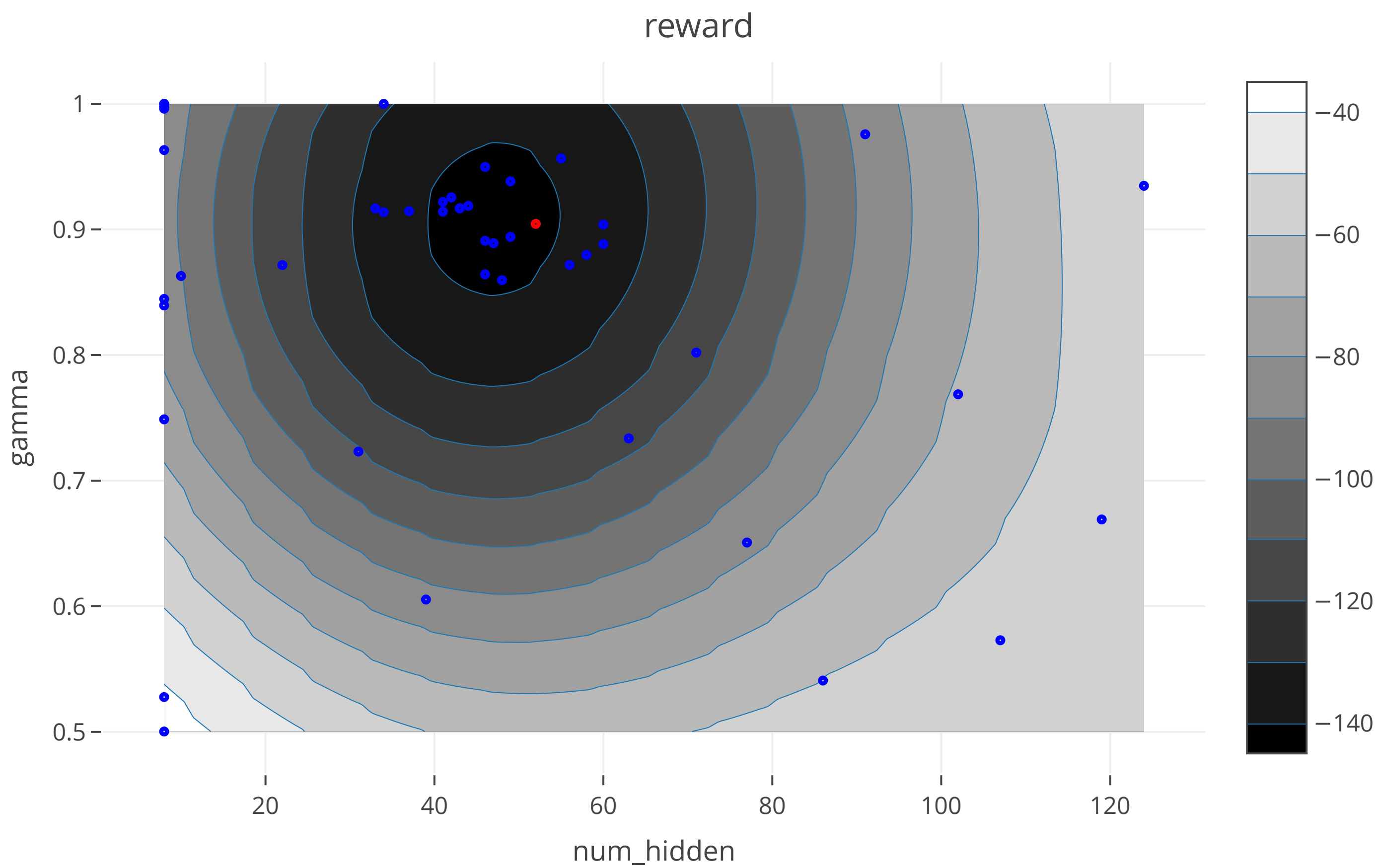
- Best solution $[52, -1.873, 0.9]$ with avg. reward ~181



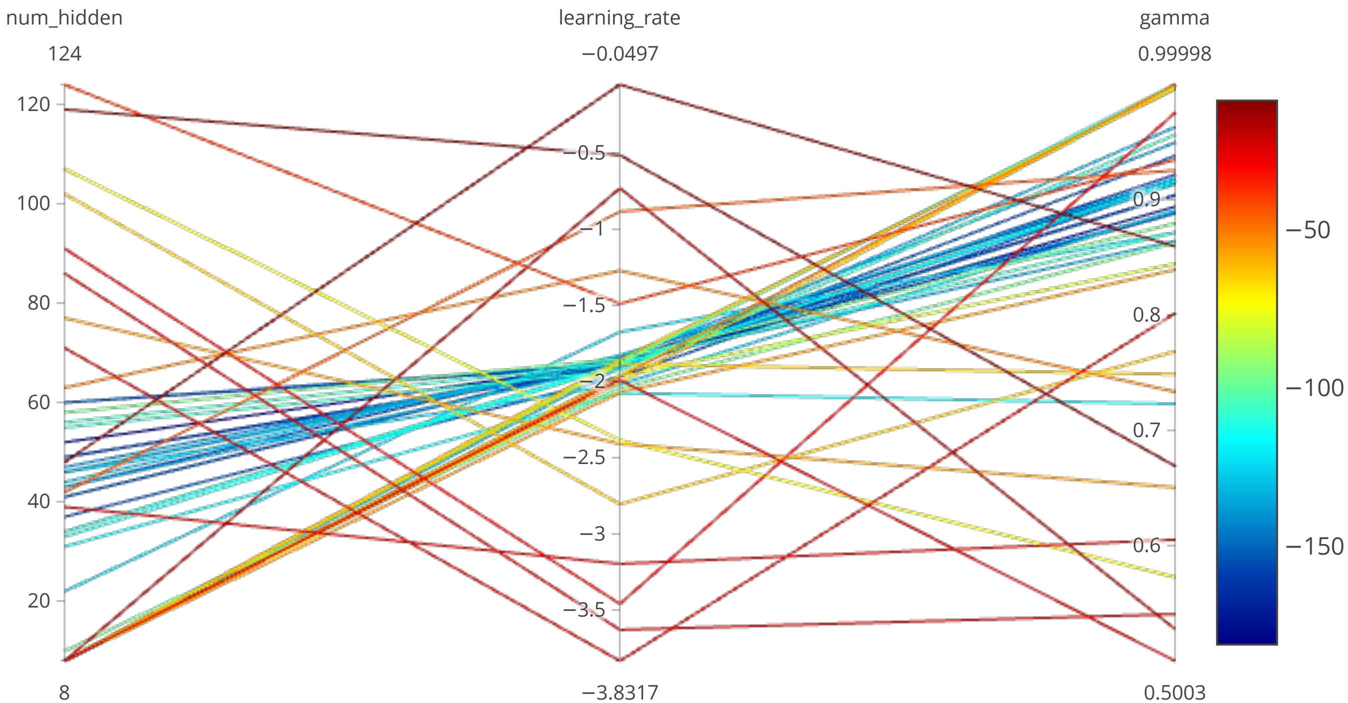
Results: reward ~ num_hidden + learning_rate



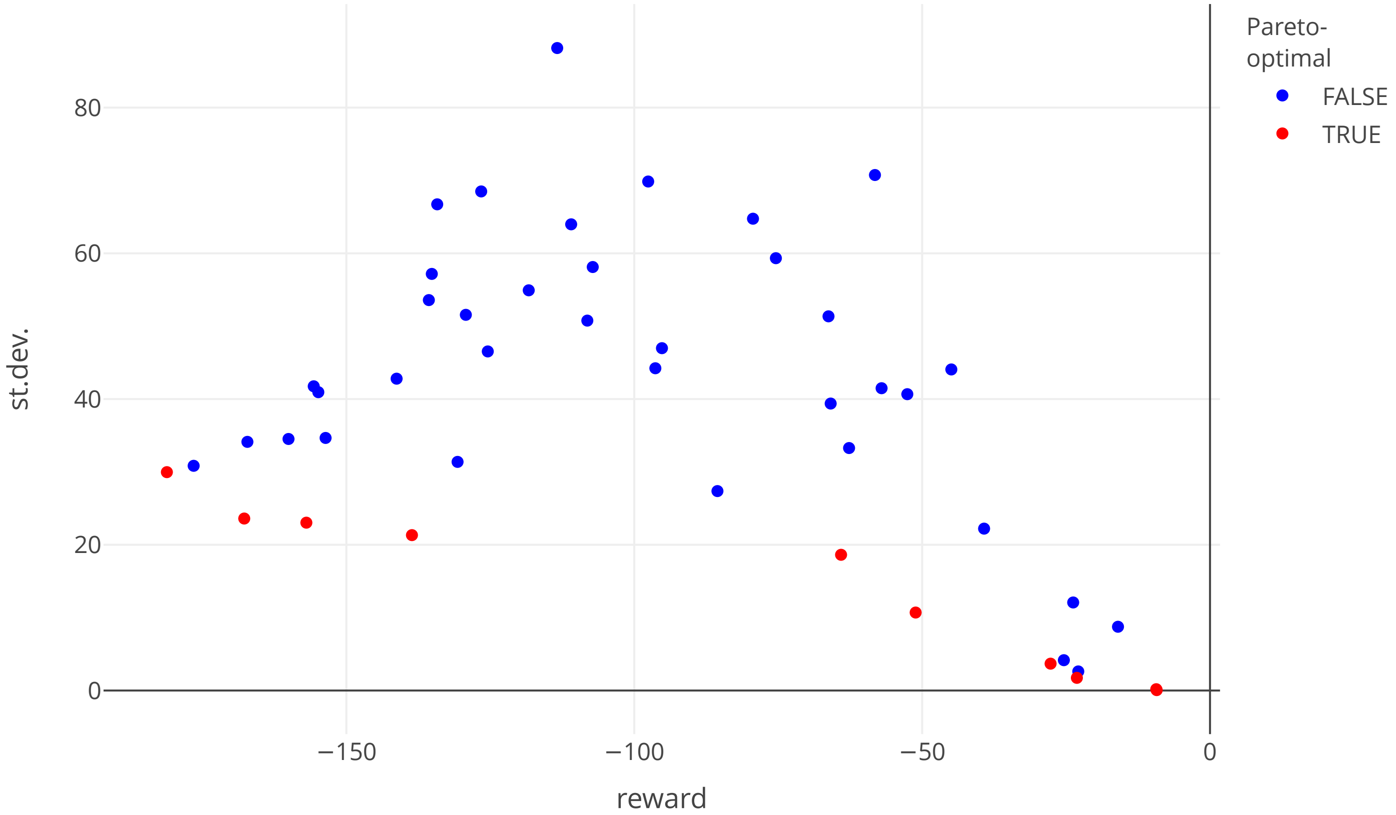
Results: reward ~ num_hidden + gamma



Results: parallel plot



Results: reward, mean vs. standard deviation



Open issues

Open issues

- Use of resources
  - How many replications do we need?
  - How long should we let the agent interact with the environment?
  - Balancing the above (they interact)?
- Multiple goals:
  - Maximize reward
  - Minimize runtime
  - Minimize memory, CPU use (e.g., for edge devices)

Open issues

- Neural architecture search
  - Representations (e.g., blocks / cells, chains, ‘unrestricted’)
  - Search operators
  - Measuring network similarities / kernels
- Conditional parameter spaces
- High number of hyperparameters
- Transfer learning

Thanks for your attention.
Questions?