TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2021

MuZero

Action Planning for Atari

Mastering Atari, Go, chess and shogi by planning with a learned model, Schrittweiser et al., Nature 2020.

While the paper describes a new system for board games (Chess, go and shogi) there is no significant improvement over AlphaZero for board games.

The real innovation here is the generalization of AlphaZero to Atari games.

I will focus on Atari Games.

Action Planning for Atari

AlphaZero uses Monte-Carlo tree search (MCTS) to "plan" into the future.

Planning requires a model of state transitions.

Q-learning and advantage actor-critic do not require the ability to plan ahead (tree search). Previous Deep-Mind systems for Atari did not use MCTS to plan ahead.

Building a Model

MuZero for Atari sees the Atari screen and rewards during play but cannot see future screens when it does MCTS to plan for future outcomes.

During MCTS MuZero uses a learned dynamic model — an RNN $g_\Phi(s, a)$ for predicting reward and the next state.

$$r^k, s^k = g_\Phi(s^{k-1}, a^k)$$

Given the RNN state model we can perform Monte Carlo tree search (MCTS) as in AlphaZero.

Training

As in AlphaZero, we train a policy network and value network.

But in MuZero we also train a model of state transitions and rewards to be used in MCTS.

At the beginning of the each MCTS the state is initialized from the observed history of screen shots and actions taken. This initialization network is also trained.

Training

The training loss function has the form

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}}\, E_{s\sim\text{Rollout}}\, \mathcal{L}^{\pi}(s) + \mathcal{L}^{V}(s) + \mathcal{L}^{R}(s) + c||\Phi||^2$$

The rollouts are generated from the actions selected by MCTS and put in a replay buffer.

The entries in the replay buffer have the form

$$s = [\ (o_1, a_1, \ldots, o_t, a_t);\ \ a_{t+1},\ r_{t+1}, \ldots, a_{t+K}, r_{t+K}, v_{t+K+1}\ ]$$

where a_t is the action taken at time t, o_t is a screen shot at time, r_t is the reward at time t (from the actual game), and v_{t+K+1} is the discounted reward encountered from time $t + K + 1$.

Training

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}}\, E_{s\sim\text{Rollout}}\, \mathcal{L}^{\pi}(s) + \mathcal{L}^{V}(s) + \mathcal{L}^{R}(s) + c||\Phi||^2$$

$$s = [\ (o_1, a_1, \ldots, o_t, a_t);\ \ a_{t+1},\ r_{t+1}, \ldots, a_{t+K}, r_{t+K}, v_{t+K+1}\ ]$$

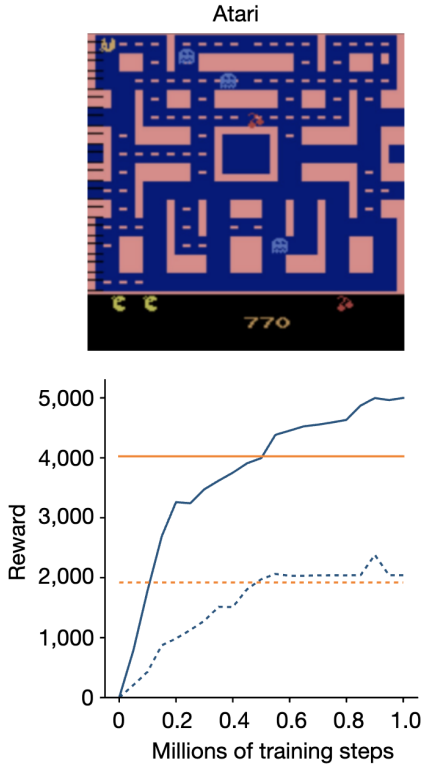
$\mathcal{L}^{\pi}$ trains the policy network to predict a_{t+1} (a cross-entropy loss).

$\mathcal{L}^{V}$ trains the value network to predict v_{t+K+1} (square loss).

$\mathcal{L}^{R}$ trains the reward network to predict each r_{t+k} (square loss).

The state dynamics network is trained implicitly by these loss functions.

Results



These are human normalized scores averaged over all 57 Atari games. The orange line is the previous state of the art system. Solid lines are average scores and dashed lines are median scores.

END