AlphaGo family

AlphaGo family

- An evolution of methods proposed by DeepMind:
  - **AlphaGo** (Silver et al. 16) where the authors describe a MCTS method with RL and self-play that learns to play Go beating the Human World Master of the game
  - **AlphaZero** (Silver et al. 17), an evolution where agent learns purely using RL without any previous knowledge of the game
  - **Muzero** (Schrittwieser et al. 19) that learns to play *without a model of the game* (model-free RL). It can be extended to any kind of problem in RL (we will see it in Model-based methods)

AlphaZero

- Simpler than AlphaGo and applicable to other games
- In AlphaZero there is only a Neural Network f_θ that outputs both, the value of a state $v_\theta(s)$ and the distribution of probabilities for each action of a stochastic policy $P_\theta(a|s)$
- It applies self-play schema together with a variation of MCTS with learning of f_θ

AlphaZero details

- It uses MCTS where:
  - Selection step done according:

$$a_t^{UCB1} = \arg\max_{a_i \in \mathcal{A}} Q\left(s_t, a_i\right) \; + \; c \; P_\theta\left(a_i | s_t\right) \frac{\sqrt{t(s_t)}}{1 + N\left(s_t, a_i\right)}$$

  - Compared with MCTS there is no simulation! **The prediction of the value of the expanded node is used to backpropagate results**
  - Action executed while self-play is according to sampling distribution:

$$\pi(s, a) = \frac{N\left(s_t, a_i\right)}{t(s_t)}$$

AlphaZero details

- Learning of f_θ is done with cases collected from the play of the kind

$$(s_t, \pi(s_t), z_t)$$

  where for each state in the trajectory s_t we store the policy distribution $\pi(s_t)$ and z is the final outcome of the trajectory (win or lose)

- Loss for f_θ is simply:

$$l = \sum_t \left(v_\theta\left(s_t\right) - z_t\right)^2 - \pi(s_t) \cdot \log\left(\vec{P}_\theta\left(s_t\right)\right)$$

- Loss minimize at the same time the prediction on final game and mismatch between policy used and the predicted by the network

AlphaZero discussion

- Some numbers for Go from a nice cheatsheet (not mine) of the paper:
  - Self-play of about 4.9 million games
  - At each iteration of SelfPlay the agent *a* plays 25.000 games against itself
  - We continue untill in the last 400 games agent *a* wins 55% of games
  - Number of iterations for growing a MTCS: 1600 simulations
  - Training of the neural network is done with batchsize 2048 from buffer containing 500.000 last games
  - In the case of go, input is 17 boards (19x19) stacked representing current and the last 7 boards per player (x2) plus a board to represent the turn
  - Neural Network if composed of 40 residual convolutional layers

AlphaZero discussion

- General algorithm for zero sum games
- Very effective and state of the art is most zero-sum games (even in chess![2]).
- No examples of playing required. Learns from scratch.
- Still needs to know the rules of the game (model of the world). Muzero solves this problem.
- When playing in production still generate a tree Why? Could not we use the policy learn? In practice better performance.
- On the dark side: Time to learn by self-play is high. Not easy to find NN architectures for each game. Large amount of resources to play (still uses MCTS)

[2] See LeelaZero