TTIC 31230 Fundamentals of Deep Learning

AlphaZero Problems.

Backgound. A version of AlphaZero can be defined as follows.

To select a move in the game, first construct a search tree over possible moves to evaluate options.

The tree is grown by running "simulations". Each simulation descends into the tree from the root selecting a move from each position until the selected move has not been explored before. When the simulation reaches an unexplored move it expands the tree by adding a node for that move. Each simulation returns the value $V_\Phi(s)$ for the newly added node s.

Each node in the search tree represents a board position s and stores the following information which can be initialized by running the value and policy networks on position s.

- $V_\Phi(s)$ — the value network value for the position s.
- For each legal move a from s, the policy network probability $\pi_\Phi(s, a)$.
- For each legal move a from s, the number $N(s, a)$ of simulations that have taken move a from s. This is initially zero.
- For each legal move a from s with $N(s, a) > 0$, the average $\hat{\mu}(s, a)$ of the values of the simulations that have taken move a from position s.

In descending into the tree, simulations select the move $\mathrm{argmax}_a\, U(s, a)$ where we have

$$U(s,a) = \left\{ \begin{array}{ll} \lambda\pi_\Phi(s,a) & \text{if } N(s,a) = 0 \\ \hat{\mu}(s,a) + \lambda\pi_\Phi(s,a)/N(s,a) & \text{otherwise} \end{array} \right. \tag{1}$$

When the search is completed, we must select a move from the root position. For this we use a post-search stochastic policy

$$\pi_{s_{\mathrm{root}}}(a) \propto N(s_{\mathrm{root}}, a)^\beta \tag{2}$$

where β is temperature hyperparameter.

For training we construct

$$\text{a replay buffer of triples } (s_{\mathrm{root}}, \pi_{s_{\mathrm{root}}}, z) \tag{3}$$

accumulated from self play where s_{root} is a root position from a search during a game, $\pi_{s_{\mathrm{root}}}$ is the post-search policy constructed for s_{root}, and z is the outcome of that game.

Training is then done by SGD on the following objective function.

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{(s,\pi,z)\sim\text{Replay},\; a\sim\pi} \left(\begin{array}{l} (V_\Phi(s)-z)^2 \\ \\ -\lambda_1 \log \pi_\Phi(a|s) \\ \\ +\lambda_2||\Phi||^2 \end{array}\right) \qquad (4)$$

Problem 1.

(a) Consider the case of $\lambda < 1$ vs. $\lambda > 1$ in (1). Which value of λ leads to more diversity in the choice of actions during different simulations? Explain your answer.

(b) Which of $\lambda < 1$ or $\lambda > 1$ in (1) is more consistent with viewing $U(s,a)$ as a form of "upper bound" in the upper confidence bound (UCB) bandit algorithm? Explain your answer.

Problem 2 We consider replacing the policy network π_Φ with a Q-value network Q_Φ so that each node s stores the Q-values $Q_\Phi(s,a)$ rather than the policy probabilities $\pi_\Phi(s,a)$. We then replace (1) with

$$U(s,a) = \begin{cases} (1+\delta)Q_\Phi(s,a) & \text{if } N(s,a)=0 \\ \hat{\mu}(s,a) + (1+\delta)Q_\Phi(s,a)/N(s,a) & \text{otherwise} \end{cases} \qquad (1')$$

and leave (2) and (3) unchanged. Rewrite (4) to train $Q_\Phi(s,a)$ by minimizing a squared "Bellman Error" between $Q_\Phi(s,a)$ and the outcome z over actions drawn from the replay buffer's stored policy. Presumably this version does not work as well.

Problem 3. We consider replacing the policy network π_Φ with an advantage network A_Φ so that each node s stores the A-values $A_\Phi(s,a)$ rather than the policy probabilities $\pi_\Phi(s,a)$. We now have each node s also store $\hat{\mu}(s)$ which equals the average value of the simulations that go through state s. We then replace (1) with

$$U(s,a) = \begin{cases} (1+\delta)(A_\Phi(s,a)+V_\Phi(s)) & \text{if } N(s,a)=0 \\ \hat{\mu}(s,a) + (1+\delta)(A_\Phi(s,a)+V_\Phi(s))/N(s,a) & \text{otherwise} \end{cases} \qquad (1'')$$

and leave (2) unchanged and replace (3) by

$$\text{a replay buffer of tuples } (s_{\text{root}}, \pi_{s_{\text{root}}}, z, \hat{\mu}(s_{\text{root}}), \hat{\mu}(s_{\text{root}}, \cdot)) \qquad (3')$$

Rewrite (4) to train $A_\Phi(s,a)$ by minimizing a squared "Bellman Error" between $A_\Phi(s,a)$ and $\hat{A}(s,a)$ defined as follows

$$\hat{A}(s,a) = \begin{cases} A_\Phi(s,a) & \text{if } N(s,a)=0 \\ \hat{\mu}(s,a) - \hat{\mu}(s) & \text{otherwise} \end{cases} \qquad (5)$$

Although a valiant attempt, this version presumably also does not work as well. It is interesting to contemplate the magic of (1) through (4) as used in AlphaZero (in spite of the seemingly ill-formed semantics of (1)).