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

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

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

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

where β is a temperature hyperparameter.

For training we construct

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

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.

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

Problem 1.

(a) If we increase λ_u encourage more diversity or less diversity in the actions selected by simulations? Explain your answer.

Solution: Increasing λ_u leads to more diversity because larger values of λ_u cause $U(s,a)$ to decline as $N(s,a)$ increases leading to the simulation moving away from a as $N(s,a)$ increases.

(b) Consider the case of very large λ_u so that the term $\lambda_u \pi_\Phi(a|s)/N(s,a) >> \hat{\mu}(s,a)$. Equivalently we can change the definition of $U(s,a)$ to be

$$U(s,a) = \frac{\pi_\Phi(a|s)}{\min(1, N(s,a))} \quad (5)$$

After running a some number of simulations from s define a^* by

$$a^* = \underset{a}{\operatorname{argmax}}\, U(s,a)$$

In other words a^* is the move that would be expanded in the next simulation visiting S. Consider a move a othr than a^*. Give a lower bound on $N(s,a)$ in terms of $U(s,a^*)$ and $\pi_\Phi(s,a)$ where $U(s,a)$ is defined by (5).

Solution:

$$\begin{aligned} U(s,a^*) &\geq U(s,a) \\ U(s,a^*) &\geq \frac{\pi_\Phi(a|s)}{N(s,a)} \\ N(s,a) &\geq \frac{\pi_\Phi(s|a)}{U(s,a^*)} \quad (6) \end{aligned}$$

Note that (6) holds for all a. So for very large λ_u the number of simulations visiting an action a is proportional to $\pi_\Phi(a|s)$ for some top k actions.

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

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

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.

Solution:

$$\Phi^* \;=\; \operatorname*{argmin}_{\Phi}\; E_{(s,\pi,z)\sim\text{Replay},\; a\sim\pi} \begin{pmatrix} (V_\Phi(s)-z)^2 \\ \\ -\lambda_1(Q(s,a)-z)^2 \\ \\ +\lambda_2||\Phi||^2 \end{pmatrix}$$

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

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

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

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

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

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

Solution:

$$\Phi^* \;=\; \operatorname*{argmin}_{\Phi}\; E_{(s,\pi,R)\sim\text{Replay},\; a\sim\pi} \begin{pmatrix} (V_\Phi(s)-z)^2 \\ \\ \lambda_A\;(A_\Phi(s,a)-\hat{A}(s,a))^2 \\ \\ +\lambda_R\;||\Phi||^2 \end{pmatrix} \qquad (2)$$

Although a valiant attempt, this version also presumably also does not work as well.