Artificial Intelligence Nanodegree

Research review (AlphaGo)

Project 2: Build a Game-Playing Agent

Antonio Ferraioli

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence because of its huge search space and the difficulty of evaluating board positions and moves. AlphaGo represents a new approach for computer Go employing convolutional neural networks. A description of the network is the following.

Training pipeline first stage: supervised learning of policy networks

The network inputs are representations of the board positions (19x19 images from expert human games). A first goal is to determine a policy $p_\sigma(a|s)$ (i.e. a probability distribution over possible moves a in position s). Input is passed through many convolutional layers with parameters σ (SL policy network); the output is p_σ represented by a probability map over the board. The policy network is trained on randomly sampled state-action pairs (s, a) using stochastic gradient ascent to maximize the likelihood of the human move a selected in state s. Additionally, a faster but less accurate rollout policy $p_\pi(a|s)$ with weights π is trained using small pattern input features.

Training pipeline second stage: reinforcement learning of policy networks

In this phase the policy network is improved by policy gradient reinforcement learning (RL). The RL policy network has the same structure of the SL policy network. The policy is p_ρ and is initialized to the SL policy network, then it is improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous randomly selected versions of the policy network. The outcome $z_t = \pm 1$ is the terminal reward at the end of the game from the perspective of the current player at timestep $t<T$ (T is the terminal timestep). Weights are updated at each timestep t in order to maximize expected outcome.

Training pipeline final stage: reinforcement learning of value networks

This stage of the training pipeline focuses on position evaluation, estimating a value function $v^p(s)$ that predicts the outcome from position s of games played using policy p for both players. $v^p(s)$ is approximated using a value network $v_\theta(s)$ with weights θ. This function is determined using a network similar to the policy network. Value network weights are trained by regression on state-outcome pairs (s, z), using stochastic gradient descent to minimize the mean squared error between the predicted value $v_\theta(s)$ and the corresponding outcome z.

Searching with policy and value networks

AlphaGo combines the policy and value networks in an MCTS algorithm. Tree is traversed by selecting the edge with maximum action value Q plus a bonus $u(P)$ that is proportional to stored prior probability P (*selection*). When the traversal reaches a leaf node, this may be expanded and processed once using p_σ; the output probabilities are stored as prior probabilities P for each action (*expansion*). The leaf node is evaluated using the value network v_θ at leaf node and by running a random rollout to the end of the game using the fast p_π policy; eventually these evaluations are combined (*evaluation*). At the end of simulation the action values and visit counts of all traversed edges are updated (*backup*). Once the search is complete, the algorithm chooses the most visited move from the root position.

Evaluating the playing strength of AlphaGo

To evaluate the playing strength of AlphaGo, an internal tournament among variants of AlphaGo and several other Go programs was run. AlphaGo resulted many *dan* ranks stronger than any previous Go program, winning 494 out of 495 games against other Go programs. In 2015, the distributed version of AlphaGo won the match against European Go championship winner Fan Hui 5 games to 0 (the first time that a computer Go program has defeated a human professional player).