Learning to repeat: Fine Grained Action Repetition for Deep reinforcement learning

Reinforcement learning is based on evaluative feedback from the environment and learning occurs using the rewards sampled from the environment and not using a set of training points like the supervised learning setup. Most of the Reinforcement learning problems can be modelled using Markov decision processes using some abstract definitions of "state" and "action". In case of finite state spaces and discrete actions, it suffices to have tabular representations for Q-values or policies. But as the number of states increases, these representations pose a problem due to memory constraints. One more problem with lookup tables is that they don't generalise well on unseen states. Similar situation occurs when the action space is not discrete. But this paper considers reinforcement learning problems with a huge state space and discretized set of actions. A notable approach in the direction of "generalisation" is Deep Q-networks, where a deep neural network is used to model Q(s, a) for all the discrete actions. In other words, the deep net captures the abstract representations of states after interacting with the environment. As successive observations (instances of the game environment) are correlated DQNs tend to produce similar feature representations for them and hence the actions taken in consecutive frames are also correlated. It was observed that a DQN agent plays the same action multiple times continuously and hence action repetition is somehow embedded in DQN setups! So, it makes sense to train the network on grounds of action selection and action repetition. Durugkar et al shows that using macro-actions (same action repeated 'k' times) results in temporal abstractions i.e the agent learns to exploit the "repetition" parameter to move from temporally distant, yet advantageous states. Also, the time scale for action repetition in all other works till now was predominantly static. But FiGAR enables any DRL algorithm to learn temporal abstractions using temporally extended macro-actions. Macro actions in essence model a semi-MDP, where actions can take varying amounts of time to complete, possibly even non-integer or randomly-distributed amounts of time. The setup is however a bit less stochastic since we now deterministically know repeat a set of actions in FiGAR. The Dynamic Frame-skip Deep-Q network highlights the the importance of the action repetition setup, but FiGAR overcomes the problem of having a large of number of parameters that would otherwise occur. There have to attempts to capture more general macro actions like in the case of STRAW, and more general parametrization of policies. Extending this further to *options* has been a long standing problem in RL, where we have a sub-policy that executes until we achieve a certain terminal state (which is different from a macro action where the set of actions is predetermined at the start of executing these set of actions).

FiGAR factorises the way in which the agent interacts. A separate policy for action-repetition (AR) is maintained in addition to the action-selection (AS) policy. For easier learning of these two policies, some parameters of the network can be tied and made common to both these output units. The paper discusses this general idea and extends this to three RL frameworks viz. A3C, TRPO and DDPG. A3C uses a parametric representation for the policy (actor) and the value function (critic), and thus the agent learns both of them. In FiGAR-A3C, we obtain a distribution over the actions (AS policy), a distribution over the repetitions (AR policy) and an estimate of V(s). From this factorisation it is clear that, instead of considering all possible combinations of (action, repetition) which will blow up the output units to |A|.|W| (same conventions as the paper), FiGAR uses just |A|+|W| output units. The next algorithm TRPO, guarantees a monotonic improvement in stochastic policies via modified surrogate loss function. The TRPO paper also extends this to work with finite sample counts and arbitrary parametrization of the policies. The surrogate loss function is not directly minimized but is subject to collection of additional constraints (which are imposed to guarantee the monotonous improvement). Deterministic policy gradient (DPG) theorem relates the gradient of a performance function of a deterministic policy in a continuous action space w.r.t its policy parameters. The DPG model consists of an actor (a policy in continuous action space) and a critic Q(s, a) to evaluate the action. DDPG is an extension of this theorem where the actor and critic are non-linear function approximators, a deep net! This is extended to FiGAR-DDPG with small modifications. We have an AR policy which is a distribution over the repetition values. These two policies are fed into the critic and gradients for the actors are derived from the critic, as is the case with DDPG. There is no explicit loss function for the actor as the critic, Q(s, a) depends on the policies directly.

An experimental setup comparing the A3C and the FiGAR-A3C shows that latter performs considerably better in a plethora of games compared to the A3C algorithm. This is specifically seen in games that are required to model temporally distant states (like Seaquest) and also when it is logical to apply action repetition. The action repetition hence allows the model to explore more states due to an increased repetition rate. However this raises a number of questions as to whether the action repetition collapses to one value and whether it makes sense to take these larger macro actions. An analysis of this highlights that this is not the case, and the distribution of action repetition rate is in fact no highly skewed. Another interesting point to note is that the FiGAR-A3C is not highly dependent on |W|. The paper also does an in-depth analysis comparing greedy and stochastic policies (sampling from the final distribution π) when obtaining the test scores. Interestingly, the performance does not considerably change. As a result, it makes more sense to choose stochastic policies over greedy policies since we can execute actions with higher repetition rates (saving computation) without drastically affecting the performance. There are multiple variants of FiGAR (all of which seem to perform well) and the appendix consider one more such variant. One such variant ignores the action repetition while generating test scores in order see whether action repetition is in fact useful and not just useful for exploration. The experiments clearly confirm this and indicate that the network has in fact encoded non-trivial temporal abstractions which are realized via the action repetitions. In the experiment using TRPO analyzing the MuJoCo domain, the performance difference does not improve drastically, possibly since action repetitions are not as useful. Moreover, parametrization of repetitions as a continuous variable may also prove to be useful and can be an area for further research. Other possible areas of research could be to look at different parametrizations of the policy, in order to capture more general macro-actions. Also most mechanisms(using macro actions) cannot change the set of actions once the macro action has started, and one other area of research would be to incorporate *options* into the Deep RL setup.