Recent Policy Gradient and Actor Critic Methods

We will be looking at four different policy gradient and actor critic methods :

- TRPO ('Trust Region Policy Optimization')
- PPO ('Proximal Policy Optimization')
- ACKTR (Kronecker-factored approximation)
- PCL ('Path Consistency Learning')

These methods are all compatible with neural network function approximation.

Trust Region Policy Optimization - 1 (Schulman et al. 2015)

- Goal = stabilizing (parameterized) policy updates.
- The high-level idea is to take steps in directions that improve the policy, while simultaneously not straying too far from the old policy.
- Making too large a change from the previous policy, especially in high-dimensional, nonlinear environments, can lead to a dramatic decrease in performance. For example, a little forward lean helps running speed, but too much forward lean leads to a crash.
- TRPO takes a principled approach to controlling the rate of policy change - the algorithm places a constraint on the average Kullback-Leibler divergence between the new and old policy after each update.

Trust Region Policy Optimization - 2

- The change in reward η by updating policy π to policy $\hat{\pi}$ is

$$\eta(\hat{\pi}) = \eta(\pi) + \sum_s \rho_{\hat{\pi}}(s) \sum_a \hat{\pi}(a|s) A_\pi(s, a) \tag{1}$$

- The dependency on $\hat{\pi}$ through $\rho_{\hat{\pi}}(s)$ is complex, so we approximate/linearize via

$$L_\pi(\hat{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \hat{\pi}(a|s) A_\pi(s, a) \tag{2}$$

- In this case, one proves that by taking linear mixture steps, we can guarantee monotonic policy improvement :
  $\pi_{new}(a|s) = (1 - \alpha)\pi_{old}(a|s) + \alpha \arg\max L_{\pi_{old}}(\pi)(a|s)$

Trust Region Policy Optimization - 3

Theorem (Schulman, 2015):

$$\eta(\hat{\pi}) \geq L_\pi(\hat{\pi}) - \frac{4\gamma}{(1-\gamma)^2} \cdot \max_{s,a}|A_\pi(s,a)| \cdot D_{KL}^{\max}(\pi, \hat{\pi})$$

(proven using policy coupling methods, and then comparing total variation distance and KL). Based on this pessimistic surrogate, we could iterately solve for policies (π_i) :

$$\pi_{i+1} = \arg\max_\pi [L_{\pi_i}(\pi) - C \cdot D_{KL}^{\max}(\pi_i, \pi)]$$

This is a penalized optimization update.

Trust Region Policy Optimization - 4

- TRPO is a robust approximation to the update on the previous slide, using a constraint on the KL divergence rather than a penalty, in order to robustly allow large updates.
- We now use a policy parameter θ so that the algorithm becomes:
- Maximize $L_{\theta_{old}}(\theta)$ over θ, subject to constraint $D_{KL}^{\max}(\theta_{old}, \theta) \leq \delta$.
- Using Monte-Carlo expectations and cheating a little, this is equivalent to

$$\max_{\theta} \quad \mathbb{E}_{s \sim \rho_{\theta_{old}}, a \sim \theta_{old}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} Q_{\theta_{old}}(s, a)]$$

- In practice the algorithm performs the optimization not by SGD, but by taking a quadratic approximation to the KL-divergence, and calculating conjugate gradients efficiently (see Hessian-vector products).

Proximal Policy Optimization - 1 (Schulman et al. 2017)

- TRPO is useful especially in continuous control tasks, but isn't easily compatible with algorithms that share parameters between a policy and value function or auxiliary losses.
- The algorithm is bespoke as it uses second-order information
- We would like to modify the policy gradients loss function in order to do a trust region update compatible with stochastic gradient descent.

Proximal Policy Optimization - 2

- This happens with the new objective function

$$L^{CLIP}(\theta) = \mathbb{E}_{empirical}\left[\min(r_t(\theta), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)) \cdot A_t\right]$$

- r_t is the ratio of the probability under the new and old policies, respectively
- ϵ is a hyperparameter around 0.1 or 0.2.
- The algorithm is simplified : no more KL penalties and adaptive updates. We can train with SGD as usual.
- Why does this work ?

Proximal Policy Optimization - 3

- For comparison the policy gradient objective is with the same notations
$$L^{PG}(\theta) = \mathbb{E}\left[\log \pi_\theta(a_t|s_t) \cdot A_t\right]$$
- The TRPO objective is, neglecting the KL term,
$$L^{TRPO}(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot A_t\right] = \mathbb{E}\left[r_t(\theta) \cdot A_t\right]$$
- In PPO's L^{CLIP}, the clipping term modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$.
- The min term in this scheme is conservative: we only ignore the change in probability ratio when it would make the objective improve, and we include it when it makes the objective worse.

ACKTR - 1 (Wu et al. 2017)

- Idea 1 = policy gradient updates following the natural gradient, gives us the direction in parameter space that achieves the largest instantaneous improvement in the objective per unit of change in the output distribution of the network - as measured using the KL-divergence.
- That is, the underlying is the Fisher metric:

$$\mathbb{E}_{\pi}\left[\nabla_{\theta}\log\pi(a_t|s_t)(\nabla_{\theta}\log\pi(a_t|s_t))^T\right]$$

- TRPO makes use of Hessian vector-products, in order to avoid inverting the Fisher information matrix explicitly.

ACKTR - 2

- K-FAC (Kronecker-Factored Approximation) is a quasi-Newton algorithm ($\theta_{t+1} \leftarrow \theta_t - \epsilon H_f^{-1} \nabla_\theta f$) replacing SGD optimizers for the training of neural networks.
- Idea 2 = use a structured approximation to the inverse Fisher matrix H_f^{-1}. This makes more progress per step, hence trades sample efficiency for computational cost.
- We can combine both natural policy gradients improvement and K-FAC.
- Helps solving the sample efficiency problem (faster per step convergence than PG+ADAM).

PCL - 1 (Nachum et al. 2017)

- Idea = make the (policy) entropic regularization in actor-critic discounted and recursive just like the rewards themselves.
- Inserting $\pi \log \pi$ recursively, this gives the Bellman equation for the loss as

$$L^{ENT}(s,\pi) = \sum_a \pi(a|s)\left[r(s,a) - \tau \log \pi(a|s) + \gamma L^{ENT}(s',\pi)\right]$$

- Doing this, all the math works out, with one-hot hard-maxes replaced by Boltzmann soft-maxes (log-sum-exp operator, because it is the Fenchel-Legendre dual of the entropy regularization functional).

$$V^*(s) = L^{ENT}(s,\pi^*) = \tau \log \sum_a \exp\left[\frac{r(s,a) + \gamma V^*(s')}{\tau}\right]$$

$$\pi^*(a|s) = \frac{\exp\left[\frac{r(s,a)+\gamma V^*(s')}{\tau}\right]}{\exp(\frac{V^*(s)}{\tau})}$$

PCL - 2

- Hence for each pair (s, s') and action a we get at optimality:

$$V^*(s) - \gamma V^*(s') = r(s, a) - \tau \log \pi^*(a|s)$$

- Therefore on any full d-length sub-trajectory of states $(s_{i,i+d})$, we can define the loss $C(s_{i,i+d}, \theta, \phi)$ as

$$-V_\phi(s_i) + \gamma^d V_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left[r(s_{i+j}, a_{i+j}) - \tau \log \pi_\theta(a_{i+j}|s_{i+j}) \right]$$

- The joint (θ, ϕ) loss

$$L_{\theta,\phi}^{PCL} = \frac{1}{2} \sum_{s_{i,i+d}} C(s_{i,i+d}, \theta, \phi)^2$$

  is the MSE objective whose gradients are minimized by PCL.
- Note that under that formulation policy gradients and (soft) Q-learning are equivalent !

DRL is hard... DRL that matters

- Encouraging reproducibility in reinforcement learning
- 'RL algorithms have many moving parts that are hard to debug, and they require substantial effort in tuning in order to get good results.'
- False Discoveries everywhere ? Dependency on the random seed
- Multiple runs are necessary to give an idea of the variability. Reporting the best returns is not enough - at the very least, report top and bottom quantiles
- Don't get discouraged !

Don't be an alchemist

Suggestions for experiment design

Factors influencing training, by order of importance:

- Fix your random seed for reproducibility !
- *Reward scaling* (and more generally reward engineering) help a lot
- *Number of steps* in the calculation of returns
- *Discount factor*, if you have one, also an issue

Clip gradients to avoid NaNs, visualize them with TensorBoard, and finally be patient waiting for convergence...

To go further - References

- Soft Actor Critic
- Boosted Dual Actor Critic
- IMPALA
- etc...