Learning Continuous Control Policies by Stochastic Value Gradients (SVG) (1510.)

- Background
  - discrete-time MDP with continuous states and actions
  - time-varying reward function
  - either finite-horizon (time-dependent value functions) or infinite-horizon (stationary value functions) sum of rewards
- Stochastic value gradients
  - Backpropagation in stochastic Bellman equation.
  - Compute gradients along real trajectories -> SVG(∞).
  - Integrate value function critics -> SVG(1), extending to infinite-horizon control.
  - In the special case of additive noise, derivatives of modelare noise independent. Otherwise, use re-parameterized generative model to infer the missing noise variables.
- SVG(∞)
  - Algorithm:
    - repeat
      - sample one length-T trajectory under π, append to $\mathcal{D}$
      - train $\hat{f}$ with $\mathcal{D}$
      - on the latest trajectory:
        - infer $\xi|\ (s, a, s')\ \text{ and } \eta|(s, a)$
        - do backward recursions from T downto 0:
          $$v_\theta = \left[r_\mathrm{a}\pi_\theta + \gamma\left(v'_{\mathrm{s}'}\hat{\mathrm{f}}_\mathrm{a}\pi_\theta + v'_\theta\right)\right]\Big|_{\eta,\xi}$$
          $$v_\mathrm{s} = \left[r_\mathrm{s} + r_\mathrm{a}\pi_\mathrm{s} + \gamma v'_{\mathrm{s}'}\left(\hat{\mathrm{f}}_\mathrm{s} + \hat{\mathrm{f}}_\mathrm{a}\pi_\mathrm{s}\right)\right]\Big|_{\eta,\xi}$$
      - update policy with gradient v_θ^0
  - On-policy.
- SVG(1)-ER
  - Algorithm:
    - repeat
      - generate transitions under π_θ, append to $\mathcal{D}$
      - train $\hat{f}$ on $\mathcal{D}$
      - train $\hat{V}$ on $\mathcal{D}$
      - sample transitions from $\mathcal{D}$ generated in previous loops
      - compute empirically estimated importance-weighting gradient (which involves value function)

        - update policy with the gradient
    - Critic reduces the variance of the gradient estimates.
    - Extends to infinite-horizon.
    - Off-policy.

Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning (1708.)

- Model-based RL
    - Dynamics function predicts state change over Δt.
    - Training the learned dynamics function:
        - collecting training data with random exploration, off-policy
        - data preprocessing: normalization and Gaussian corruption
        - train the model with H-step validation errors (multi-step open-loop predictions)
        - random-sampling shooting with MPC, use model to estimate optimal action sequence and execute the first action
    - Algorithm:
        - Gather random trajectories ($\mathcal{D}_{\text{RAND}}$)
        - repeat
            - update $\hat{f}_\theta$ with gradient descent on MSE ($\mathcal{D}_{\text{RAND}}$ and $\mathcal{D}_{\text{RL}}$)
            - model-based MPC controller gathers T new on-policy (real) transitions ($\mathcal{D}_{\text{RL}}$)
- MB-MF:
    - MF learner initialization
        - gather example trajectories with MPC controller with pre-learned $\hat{f}$
        - train a neural network Gaussian policy with leanred mean and fixed covariance to match expert trajectories via SGD on MSE
        - apply DAGGER
    - MF RL
        - TRPO

Model-Based Reinforcement Learning via Meta-Policy Optimization (MB-MPO) (1809.)

- Meta-RL learns an initialization θ^* such that for any task $\mathcal{M}_k \sim \rho(\mathcal{M})$ the policy attains maximum performance in the respective task after one policy gradient step.

  $$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathcal{M}_k \sim \rho(\mathcal{M}), f_k, \pi_{\theta'}} \left[\sum_{t=0}^{H-1} r_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right] \quad \text{s.t.: } \boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathbb{E}_{f_k, \pi_\theta} \quad \left[\sum_{t=0}^{H-1} r_k\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$$
- MB-MPO
    - Model learning: models are decorrelated with random initializatoin and different subsets of random samples; dynamic models retrained via MPC controller with warm starts

- Meta-RL on learned models
    - meta-objective:
      $\max_{\boldsymbol{\theta}} \frac{1}{K} \sum_{k=0}^{K} J_k\left(\boldsymbol{\theta}_k'\right) \quad \text{s.t.:} \quad \boldsymbol{\theta}_k' = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta})$
      where $J_k(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}\left[\sum_{t=0}^{H-1} r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) | \boldsymbol{s}_{t+1} = \hat{f}_{\boldsymbol{\phi}_k}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right]$
    - use imaginary trajectories, off-policy
    - TPRO for maximizing meta-objective, VPG for adaptation objectives $J_k(\boldsymbol{\theta})$
- Algorithm:
    - initialize policy π_θ, models $\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, \ldots, \hat{f}_{\phi_K}$
    - repeat
        - eample real trajectories with adapted policies $\pi_{\boldsymbol{\theta}_1'}, \ldots, \pi_{\boldsymbol{\theta}_K'}$, append to $\mathcal{D}$
        - train all models with $\mathcal{D}$
        - for all models do
            - sample imaginary trajectories $\mathcal{T}_k$ from $\hat{f}_{\phi_k}$ using π_θ
            - update adapted policy with $\mathcal{T}_k$
            - sample imaginary trajectories $\mathcal{T}_k'$ from $\hat{f}_{\phi_k}$ using π_{θ_k}'
        - update θ with $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \beta \frac{1}{K} \sum_k \nabla_{\boldsymbol{\theta}} J_k\left(\boldsymbol{\theta}_k'\right)$ using $\mathcal{T}_k'$

Proximal Policy Optimization Algorithms (PPO) (1707.)

- Proposed objective function enables multiple epochs of minibatch updates.
- Background
    - Policy Gradient
        - estimator $\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta\left(a_t | s_t\right) \hat{A}_t\right]$
    - Trust Region
        - $$\begin{array}{ll} \underset{\theta}{\operatorname{maximize}} & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta\left(a_t | s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t | s_t\right)} \hat{A}_t\right] \\ \text{subject to} & \hat{\mathbb{E}}_t\left[\mathrm{KL}\left[\pi_{\theta_{\text{old}}}\left(\cdot | s_t\right), \pi_\theta\left(\cdot | s_t\right)\right]\right] \leq \delta \end{array}$$
        - approximately solved using conjugate gradient algorithm after linear approximation to objective and quadratic approximation to constraint
- Surrogate objective
    - clipped surrogate objective $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta) \hat{A}_t, \operatorname{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t\right)\right]$
    - adaptive KL penalty coefficient
        - $L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta\left(a_t | s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t | s_t\right)} \hat{A}_t - \beta \mathrm{KL}\left[\pi_{\theta_{\text{old}}}\left(\cdot | s_t\right), \pi_\theta\left(\cdot | s_t\right)\right]\right]$

          optimized via several epochs of minibatch SGD
        - $d = \hat{\mathbb{E}}_t\left[\mathrm{KL}\left[\pi_{\theta_{\text{old}}}\left(\cdot | s_t\right), \pi_\theta\left(\cdot | s_t\right)\right]\right]$

if $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$; if $d > d_{\text{targ}}/1.5$, $\beta \leftarrow \beta \times 2$

- If policy and value function share parameters, maximize CLIP + VF + S where CLIP = clipped surrogate objective, VF = squared-error loss for value function, S = entropy term to encourage exploration.
- Proximal Policy Optimization (PPO), AC Style
  - Algorithm:
    - repeat
      - for all N actors do
        - collect on-policy real transitions for T timesteps
        - compute advantatge estimate $\hat{A}_1, \ldots, \hat{A}_T$ (to compute loss)
      - update policy via gradients from surrogate loss L , with K epochs and minibatch size $M \leq NT$

Model-Ensemble Trust-Region Policy Optimization (ME-TRPO) (1802.)

- Vanilla Modle-Based Deep RL
  - Algorithm:
    - repeat
      - collect real samples, add to $\mathcal{D}$
      - train $\hat{f}$ with $\mathcal{D}$
      - repeat
        - collect fictitious samples with $\hat{f}$
        - BPTT
        - estimate expected sum of rewards
- Model Ensemble Trust Region Policy Optimization (ME-TRPO)
  - Algorithm:
    - repeat
      - collect real samples, add to D
      - train all K models with D
      - for each model in the ensemble (in parallel) # optimize policy using all models
        - collect fictitious samples with model
        - update policy using TRPO
        - estimate expected sum of rewards

Model-Based Value Expansion for Efficient Model-Free Reinforcement Learning (MVE) (1803)

- MVE is a value estimate for a policy π by assuming the model is accurate in depth H

- One sufficient condition under which MVE MSE <= underlying critic MSE: given an arbitrary off-policy distribution of state-action pairs β, set $\nu = \mathbb{E}\left[(f^\pi)^T \beta\right]$ where $T \sim \text{ Uniform } \{0, \cdots, H-1\}$.
- $\mathrm{MSE}_\nu(\hat{V})$ -> Bellman error w.r.t. υ -> using target $\hat{V}_k\left(\hat{s}_T\right)$ which is equivalent to train $\hat{V}$ with imaginary TD-k error. The target is used to train $\hat{V}$ on the entire support of ν instead of just β. Sampling transitions from ν is equivalent to sampling from any point up to H imagined steps into the future, when starting from a state sampled from β.
- Algorithm: MVE
    - initialize targets $\theta' = \theta$ for π, $\varphi' = \varphi$ for Q
    - initialize replay buffer $\beta \leftarrow \emptyset$
    - repeat
        - collect transitions, add to β (empirical distribution of transitions observed in real world)
        - fit dynamics $\hat{f}$ with LSE
        - repeat
            - sample $\tau_0 \sim \beta$
            - update θ with $\nabla_\theta \ell_{\text{ actor }}\left(\pi_\theta, Q_\varphi, \tau_0\right)$
            - imagine future transitions τ_t up to H
            - update φ with $\nabla_\varphi \sum_t \ell_{\text{critic}}^{\pi_{\theta'}, \hat{Q}_{H-t}}\left(\varphi, \tau_t\right)/H$ # ν-based Bellman error of Q_φ proxied by k-step MVE of $Q_{\varphi'}$
            - update targets θ', φ' with some decay

Algorithmic Framework For Model-based Deep RL with Theoretical Guarantees (1807.)

- Meta-Algorithm for MB RL with some designed discrpancy bound D and distance function d
    - For $k = 0 \ldots T$

      $$\begin{aligned} \pi_{k+1}, M_{k+1} = & \underset{\pi \in \Pi, M \in \mathcal{M}}{\operatorname{argmax}} V^{\pi, M} - D_{\pi_k, \delta}(M, \pi) \\ & \text{s.t. } d\left(\pi, \pi_k\right) \leq \delta \end{aligned}$$
    - It iteratively optimizes the lower bound over the policy π_{k+1} and the model M_{k+1}, subject to the constraint that the policy is not very far from the reference policy π_k obtained in the previous iteration.
    - The policy performance in the real environment is non-decreasing under the assumption that the real dynamics belongs to the parameterized family $\mathcal{M}$
- The meta-algorithm is instantiated by Stochastic Lower Bound Optimization (SLBO)
    - Algorithm:
        - repeat

                - repeat
                    - collect real samples, add to $\mathcal{D}$
                    - repeat
                        - repeat
                            - update $\hat{f}$ with SGD on H-step loss
                        - repeat
                            - collect fictitious samples under $\hat{f}$ as $\mathcal{D}'$
                            - optimize π_θ on $\mathcal{D}'$ by TRPO with entropy regularization

Model-Predictive Policy Learning with Uncertainty Regularization for Driving in Dense Traffic (1901.)

- Model-predictive policy learning with uncertainty regularization (MPUR)
    - action-conditional forward model
        - per-sample loss:
          $$\mathcal{L}\left(\theta, \phi; s_{1:t}, s_{t+1}, a_t\right) = \left\|s_{t+1} - f_\theta\left(s_{1:t}, a_t, z_t\right)\right\|_2^2 + \beta D_{KL}\left(q_\phi\left(z|s_{1:t}, s_{t+1}\right)\|p(z)\right)$$
        - apply z-dropout to decouple latent variables and variation in prediction model outputs due to actions
    - training policy network with uncertainty minimization (MPUR) / expert retularization (MPER)
        - use forward model to train policy network π_ψ
            - sample initial state sequence $s_{1:t}$ from the training set
            - unroll forward model over T time steps
            - backpropagation w.r.t. ψ
        - objective: minimize policy cost + uncertainty cost
          $$\underset{\psi}{\operatorname{argmin}}\left[\sum_{i=1}^{T} C\left(\hat{s}_{t+i}\right) + \lambda U\left(\hat{s}_{t+i}\right)\right], \text{ such that: } \begin{cases} z_{t+i} \sim p(z) \\ \hat{a}_{t+i} \sim \pi_\psi\left(\hat{s}_{t+i-1}\right) \\ \hat{s}_{t+i} = f\left(\hat{s}_{t+i-1}, \hat{a}_{t+i}, z_{t+i}\right) \end{cases}$$