1 BLR and GP

1.1 Properties of Multi-variate Gaussian

- $\mu_{A|B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\mathbf{x}_B - \mu_B)$.
- $\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$.
- $MX \sim \mathcal{N}(M\mu_X, M\Sigma_X M^T)$.
- Assume $p = \mathcal{N}(\mu_0, \Sigma_0)$ and $q = \mathcal{N}(\mu_1, \Sigma_1)$, then $KL(p\|q) = \frac{1}{2}(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) - d + \ln(|\Sigma_1|/|\Sigma_0|))$.
- Entropy $H(q) = -\int q(\theta)\log q(\theta)d\theta$. $H(\mathcal{N}(\mu,\Sigma)) = \frac{1}{2}\log|2\pi e\Sigma|$.

A general n-dim distribution has at least $O(2^n)$ parameters, but a Gaussian only has $O(n^2)$.

1.2 Bayesian Linear Regression

Idea: use a prior $p(w)$ on weights for the model $y = w^T x + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Then use the full posterior $p(w \mid X, Y)$ to do inference. This allows us to use a distribution as the prediction, thus quantifies the uncertainty.

If $p(w) = \mathcal{N}(0, \sigma_p^2 I)$, then MAP estimate $\mathbf{argmax}_w P(w \mid X, Y) = \mathbf{argmin}_w \left\|Y - W^T x\right\|^2 + \frac{\sigma_n^2}{\sigma_p^2}\|w\|_2^2$, i.e., Ridge Regression. Instead, BLR predicts $p(y^* \mid X, Y, x^*) = \mathcal{N}(\overline{\mu}^T x^*, x^{*T}\overline{\Sigma}x^* + \sigma_n^2)$, where $\overline{\mu} = (X^T X + \frac{\sigma_n^2}{\sigma_p^2} I)^{-1} X^T Y$, $\overline{\Sigma} = \left(\frac{\sigma_p^2}{\sigma_n^2} X^T X + I\right)^{-1}$. $x^{*T}\overline{\Sigma}x^*$ is called *epistemic* (the uncertainty about f^*) and σ_n^2 is called *aleatoric* (the uncertainty about $y^* \mid f^*$).

1.3 Gaussian Process

Goal: predict for infinite number of x^*, e.g., predict $y(x)$ for $x \in (0, 1)$.

Def: A GP is a set of random variables $f(X)$, defined on some index set X, s.t. $\exists \mu : \mathcal{X} \to \mathcal{R}$, $k : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ such that $\forall A = \{x_1, \ldots, x_m\} \subset X$, it holds that $f_A \sim \mathcal{N}(\mu_A, K_{AA})$.

Kernels: (1) symmetric and (2) positive semi-definite. Called *stationary* if $k(x, x') = k(x - x')$. Called *isotropic* if $k(x, x') = k(\|x - x'\|^2)$. Squared exponential kernel is analytic, exponential kernel is continuous but nowhere differentiable, Matren kernel with parameter v is v times differentiable.

Suppose $f \sim GP(\mu, k)$ and we observe $y_i = f(x_i) + \epsilon_i$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Then $(f \mid A, y(A)) \sim GP(\mu', k')$, where $\mu'(x) = \mu(x) + k_{x,A}(K_{AA} + \sigma^2 I)^{-1}(y_A - \mu_A)$ and $k'(x, x') = k(x, x') - k_{x,A}(K_{AA} + \sigma^2 I)^{-1} k_{A,x'}$.

The convention: set $\mu(x) = 0$ for prior. We use MLE to estimate the parameters of the kernel function, i.e., $\theta = \mathbf{argmax} p(y \mid x, f_\theta) = \mathbf{argmax} \mathcal{N}(0, K_y) = \mathbf{argmin} y^T K_y^{-1} y + \log|K_y|$, where $K_y = K_{xx}(\theta) + \sigma_n^2 I$.

1.4 Fast GP Methods

Idea: avoid the inversion of $n \times n$ matrix which is $\Theta(n^3)$.

1. Exploiting parallelism: use GPU to speed up the matrix operation. Do not change the complexity.
2. Local methods: to predict at x, only condition on "close" points x' where $|k(x, x')| > \tau$. Still expensive if many points are considered.
3. Approximate the kernel: use a low dimensional approximation $k(x, x') \approx \phi(x)^T \phi(x)$, $\phi(x) \in \mathcal{R}^m$. The complexity is $O(nm^2 + m^3)$. This includes:
   - Random Fourier Features. The Fourier transform of stationary kernels is $k(x - x') = \int p(w) e^{iw^T(x-x')} dw$, where $p(w)$ is non-negative because of the positive semi-definiteness. We can scale $p(w)$ s.t. $p(w)$ is a distribution. Then $k(x - x') = \mathbb{E}_{w,b}(z_{w,b}(x) \cdot z_{w,b}(x'))$, where $z_{w,b}(x) = \sqrt{2}\cos(w^T x + b)$, $w \sim p(w)$ and $b \sim U([0, 2\pi])$. Then use sample average as the approximation of the expectation. RFF approximates the kernel function uniformly well.
   - Inducing Point Method. Idea: summarize function via a set of inducing points u. Approximate $p(f, f^*)$ by $\int q(f^* \mid u) q(f \mid u) p(u) du$. Complexity is $O(n|u|^3)$.

2 Bayesian Inference via Approximation

Idea: approximate the exact distribution to speed up Bayesian inference which requires integration. $p(\theta \mid y) = p(\theta, y)/Z$. We assume $p(\theta, y) = p(\theta)p(y \mid \theta)$ is easy to evaluate, but $Z = \int p(\theta, y) p(\theta)$ is intractable. Therefore, we seek $p(\theta \mid y) \approx q(\theta \mid \lambda)$.

2.1 Laplacian Method

Idea: use Gaussian to approximate, estimated by second order expansion. $q(\theta) = \mathcal{N}(\hat{\theta}, \lambda^{-1})$, where $\hat{\theta} = \mathbf{argmax} p(\theta \mid y)$ and $\lambda = -\nabla^2 \log p(\hat{\theta} \mid y)$.

Problem: it first search for MAP and then matches the curvature. Could lead to poor approximation when there are multiple peaks.

2.2 Variational Inference

Idea: minimize the distance (KL divergence) between approximation and the exact distribution, i.e., $q_{\lambda^*} = \mathbf{argmin}_\lambda KL(q_\lambda \| p) = \mathbf{argmin} \int q(\theta) \log(q(\theta)/p(\theta)) d\theta$.

Minimizing KL divergence via maximizing ELBO

By expanding, $\mathbf{argmin}_q KL(q(\theta)\|p(\theta \mid y)) = \mathbf{argmax}_q \mathbb{E}_{\theta\sim q(\theta)}[\log p(y \mid \theta)] - KL(q(\theta)\|p(\theta))$, which converts the posterior to conditional evidence and prior (the ELBO). It is called Evidence Lower Bound because $\log p(y) = \log \mathbb{E}_{\theta\sim q(\theta)}[p(y \mid \theta)\frac{p(\theta)}{q(\theta)}] \geq \mathbb{E}_{\theta\sim q(\theta)}[\log(p(y \mid \theta)\frac{p(\theta)}{q(\theta)})] = \mathbb{E}_{\theta\sim q(\theta)}[\log p(y \mid \theta)] - KL(q(\theta)\|p(\theta))$, thus a lower bound of $\log p(y)$.

2.3 Markov Chain Monte Carlo

Idea: use empirical distribution (samples) as the distribution approximation, i.e., $p(y^* \mid x^*, X, Y) = \mathbb{E}_{\theta\sim p(\theta|X,Y)} p(f^* \mid x^*, \theta) \approx \frac{1}{m}\sum_{i=1}^m p(y^* \mid x^*, \theta_i)$, where $\theta_i \sim p(\theta \mid X, Y)$. The approximation deviation decreases exponentially w.r.t. #samples. Challenge: how to sample from the exact distribution.

A Markov Chain has a unique stationary distribution $\frac{1}{Z}Q(x)$ if it is ergodic (irreducible and aperiodic) and $\forall x, x'$, $Q(x)P(x' \mid x) = Q(x')P(x \mid x')$ (sufficient, unnecessary).

Metropolis-Hastings Sampling

Goal: sample when only a function proportional to the distribution is known.

Given a proposal distribution $R(x' \mid x)$, the Markov Chain is constructed as follows: for every step, sample $x' \sim R(x' \mid x_t)$; with probability $\alpha = \min\{1, \frac{Q(x')R(x_t|x')}{Q(x)R(x'|x_t)}\}$, set $x_{t+1} = x'$, o.w. set $x_{t+1} = x_t$. If $Q(x')R(x_t \mid x') = 0$, then write $\alpha = 0$, regardless of the denominator.

- MALA (aka LMC): $R(x' \mid x) = \mathcal{N}(x - \tau\nabla f(x), 2\tau I)$. For log-concave distributions ($p = \frac{1}{Z}\exp(-f(x))$ where f is convex), mixing time is polynomial.
- SGLD: SGD + Gaussian noise. Converge if $\eta_t = \Theta(t^{-1/3})$.
- HMC: add momentum.

Gibbs Sampling

Goal: sample from high-dimension when the conditional sampling is easy.

The Markov Chain is constructed as follows: for every step, randomly pick one dimension i, then update $x_i \sim P(x_i \mid x_{-i})$. In practice, we can sequentially sample all dimensions one by one. The rationale: $P(x_i \mid x_{-i}) \propto Q(x_i, x_{-i})$.

Convergence of MCMC

Ergodic Theorem: asymptotic time average (mean of function values from the ergodic MC after infinite steps) is equal to the space average (the true expectation) almost surely if the space average is finite.

In practice: ignore the first t_0 samples, i.e., burn-in period.

3 Bayesian Deep Learning

Idea: apply Bayesian methods on NN, i.e., use the posterior distribution of weights of NN to do inference. $p(y \mid x; \theta) = \mathcal{N}(\mu(x; \theta), \sigma^2(x; \theta))$.

3.1 MAP inference for BNN

If we apply $p(\theta) = \mathcal{N}(0, \sigma_p^2 I)$, then $\hat{\theta} = \mathbf{argmin} -\log p(\theta) - \sum \log p(y_i \mid x_i, \theta) = \mathbf{argmin} \frac{1}{\sigma_p^2}\|\theta\|_2^2 + \sum[\frac{1}{\sigma(x_i;\theta)^2}\left\|y_i - \mu(x_i;\theta)\right\|^2 + \log\sigma(x_i;\theta)^2]$. Therefore, the network can attenuate the loss for certain data points by attributing the error to large variance.

3.2 Variational Inference for BNN

Idea: apply variational inference on $p(\theta \mid X, Y) \approx q_\lambda(\theta)$.

$p(y^* \mid x^*, X, Y) \approx \mathbb{E}_{\theta\sim q_\lambda}[p(y^* \mid x^*, \theta)] \approx \frac{1}{m}\sum_{i=1}^m p(y^* \mid x^*, \theta_i)$. Therefore, $\mathbb{E}(y^* \mid x^*, X, Y) = \frac{1}{m}\sum_i \mu(x^*; \theta_i)$ and $\mathbf{Var}(y^* \mid x^*, X, Y) = \mathbb{E}_{\theta|X,Y}(\mathbf{Var}(y^* \mid x^*, \theta)) + \mathbf{Var}_{\theta|X,Y}(\mathbb{E}(y^* \mid x^*, \theta)) = \frac{1}{m}\sum_i \sigma^2(x^*, \theta_i) + \frac{1}{m}\sum_i(\mu(x^*, \theta_i) - \overline{\mu(x^*)})$. $\mathbb{E}_{\theta|X,Y}(\mathbf{Var}(y^* \mid x^*, \theta))$ is the aleatoric uncertainty and $\mathbf{Var}_{\theta|X,Y}(\mathbb{E}(y^* \mid x^*, \theta))$ is the epistemic uncertainty.

4 Bayesian Data Collection

Goal: find the position to collect data to get the most useful information.

4.1 Active Learning

Goal: the underlying function $\mu(x)$, i.e., maximize information gain between the dataset and the function.

Mutual Information: $I(X; Y) = H(X) - H(X \mid Y)$, where $H(X) = \mathbb{E}_X - \log p(X)$ and $H(X \mid Y) = \mathbb{E}_Y H(X \mid Y)$. It is symmetric, i.e., $I(X; Y) = I(Y; X)$. When $X \sim \mathcal{N}(\mu, \Sigma)$, $Y = X + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, $I(X; Y) = \frac{1}{2}\log|I + \sigma^{-2}\Sigma|$. The information gain $F(S)$ is the mutual information between current model and the new data points S. $F(S)$ is monotone submodular: $\forall x$ and $\forall A \subset B$, we have $F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B)$.

In general, information gain is NP-hard to optimize. Therefore, we take a greedy strategy, i.e. use the data point that maximizes the info-gain sequentially. $x_{t+1} = \mathbf{argmax} \frac{1}{2}\log(1 + \sigma_t^2(x)/\sigma_n^2) = \mathbf{argmax} \sigma_t^2(x)$, i.e., this is equivalent to pick the position with the maximum variance currently. This provides a $1 - \frac{1}{e}$ fac-

tor of guarantee of the optimal. In the heteroscedastic case, where σ_n^2 depends on x, $x_{t+1} = \textbf{argmax}\, \sigma_t^2(x)/\sigma_n^2(x)$.

4.2 Bayesian Optimization

Goal: the maximum of a function, i.e., find $\textbf{argmax}\, f(x)$.

Settings: given unknown function f, choose $\{x_i\}$ adaptively and observe $y_t = f(x_t) + \epsilon_t$ to find $x^* = \textbf{argmax}\, f(x)$. Define cumulative regret $R_T = \sum_{t=1}^{T}(\max f(x) - f(x_t))$, then sublinear R_T ($R_T/T \to 0$) is equivalent to $\max_t f(x_t) \to f(x^*)$. We use GP to model the belief about the function: $GP(\mu_t(x), \sigma_t^2(x))$.

Upper Confidence Sampling (GP-UCB)

Idea: optimism. Choose $x_{t+1} = \textbf{argmax}\, \mu_{t-1}(x) + \beta_{t+1}\sigma_t(x)$. The cumulative regret is $O\left(\sqrt{\gamma_T/T}\right)$ up to a log factor if β_t is chosen correctly, where $\gamma_T = \max_{|S|\le T} I(f; y_S)$. The regret depends on how quickly we can gain information. If we can gain information quickly, then γ_T/T decays quickly, thus the regret is small.

γ_T for common kernels (all sublinear):

1. Linear: $\gamma_T = O(d \log T)$.
2. Squared exponential: $\gamma_T = O(\log^{d+1}(T))$.
3. Matern with $v > 0.5$: $O\left(T^{\frac{d}{2v+d}}(\log T)^{\frac{2v}{2v+d}}\right)$.

Other acquisition functions include expected improvement and probability of improvement.

Thompson Sampling

Idea: draw a sample as the acquisition fucntion. At each iteration, draw $\hat{f} \sim P(f \mid x_{1:t}, y_{1:t})$ and choose $x_{t+1} = \textbf{argmax}\, \hat{f}(x)$.

5 Probabilistic Planning

ϵ-optimal policy: a policy π s.t. $|V^\pi - V^*| \le \epsilon$ for all initial settings.

5.1 Markov Decision Process

Setting: a Markov Chain where a reward function $r(x,a)$ and a transition probability $P(x' \mid x,a)$ is given for taking action a. The policy can be determined $\pi : X \to A$ and randomized $\pi : X \to P(A)$. The goal is to maximize the expected reward $J(\pi) = \mathbb{E}(\sum_{i=1}^{\infty} \gamma^{i-1} r(X_i, \pi(X_i)))$.

Value Function of A Policy

Define the value function: $V^\pi(x) = J(\pi \mid X_0 = x)$. By $V^\pi(x) = r(x,\pi(x)) + \gamma \sum_{x'} P(x' \mid x, \pi(x)) V^\pi(x')$, we can solve for $V^\pi(x)$ via solving linear equations, which is $O(|X|^3)$. For computational reasons, we can use fixed-point iteration to solve approximately: $V_t^\pi = r^\pi + \gamma T^\pi V_{t-1}^\pi$.

Policy Iteration

Bellman Theorem: the optimal policy is the greedy policy w.r.t. its induced value function. The policy iteration algorithm iteratively computes the value function of the current policy (initialized randomly) and then set the next policy to be the greedy policy of the current value function. It is guaranteed that $V^{\pi_{t+1}}(x) \ge V^{\pi_t}(x)$ and it converges to the optimal π^* in $O(n^2 m/(1-\gamma))$, up to a log factor.

Value Iteration

Idea: directly use fixed-point iteration on $V^*(x)$. At each iteration, set $Q(x,a) = r(x,a) + \gamma \sum_{x'} P(x' \mid x,a) V_{t-1}(x')$ and $V_t(x) = \max_a Q_t(x,a)$. After convergence, choose the greedy policy w.r.t. the value function.

Partially Observable MDP

Instead of observing X_t directly, we are given noisy observation Y_t of X_t. Therefore, we put a belief on states $b_t(x) = P(X_t \mid Y_{1:t})$. The transition model becomes $P(Y_{t+1} = y \mid b_t, a_t) = \sum_{x,x'} b_t(x) P(x' \mid x, a_t) P(y \mid x')$ and $b_{t+1}(x') = \frac{1}{Z}\sum_x b_t(x) P(X_{t+1} = x' \mid X_t = x, a_t) P(y_{t+1} \mid x')$.

5.2 Reinforcement Learning

Goal: learn a policy when no model about the environment is given. The agent gets information after an action.

Model-based RL

Idea: learn the underlying MDP and use the policy learned for the MDP. $\hat{P}(X_{t+1} \mid X_t, A) = \text{count}(X_{t+1}, X_t, A)/\text{count}(X_t, A)$. $\hat{r}(x,a) = \frac{1}{|S|}\sum_S R_t$, where $S = \{t : X_t = x, A_t = a\}$.

Challenge: balance the estimation of the MDP and the cumulative regret (explore-exploit dilemma).

Algorithms:

1. Temporal Difference Learning to compute value function of a policy: Bootstrap + Robbins-Monro on $V^\pi(x) = r(x,\pi(x)) + \gamma V^\pi(x')$, i.e., (1) follow π to obtain trajectories (x,a,r,x') (2) update by bootstrapping from the trajectory $\hat{V}^\pi(x) \leftarrow (1-\alpha_t)\hat{V}^\pi(x) + \alpha_t\left(r + \gamma \hat{V}^\pi(x')\right)$. Can be converted to be off-policy via replacing $V^\pi(x)$ by $Q^\pi(x,a)$: $\hat{Q}^\pi(x,a) \leftarrow (1-\alpha_t)\hat{Q}^\pi(x,a) + \alpha_t\left(r + \gamma \hat{Q}^\pi(x', \pi(x'))\right)$.
2. ϵ_t greedy: with probability ϵ_t, pick randomly, o.w. pick the best action using current MDP model. If $\sum \epsilon_t = \infty$ and $\sum \epsilon_t^2 < \infty$, then guaranteed to converge to the optimal almost surely. Cons: doesn't quickly eliminate clearly suboptimal actions.
3. $R_{\max}$: (1) add a "fairy tale" state x^*, set $r(x,a) = R_{\max}$ and $P(x^* \mid x,a) = 1$ for all states and actions; (2) at each iteration, execute the current optimal policy; (3) after collecting "enough" data, update the MDP model. It is guaranteed that (1) after a fixed timesteps, the algorithm either obtains near-optimal reward, or visits at least one unknown state-action pair; (2) with probability $1-\delta$, $R_{\max}$ reaches an ϵ-optimal policy in #steps polynomial in $|X|, |A|, 1/\epsilon, \log(1/\delta)$ and $R_{\max}$.

Cons: (1) need to store $\hat{P}(X_{t+1} \mid X_t, A)$ and $\hat{r}(x,a)$, which is $O(|X|^2|A|)$; (2) need to solve another MDP using policy/value iteration after an update.

Model-free RL

Idea: estimate V^* and use the greedy policy.

Methods:

1. Q-learning: generalized from TD-learning. After observing a transition (x,a,r,x') which does *not* necessarily follow a policy, we update $\hat{Q}^*(x,a) \leftarrow (1-\alpha_t)\hat{Q}^*(x,a) + \alpha_t\left(r + \gamma \max_{a'} \hat{Q}^*(x',a')\right)$. It is guaranteed that if all state-action pairs are chosen infinitely often and Robbins-Monro condition is satisfied, then $\hat{Q}^*$ converges to Q^* almost surely. Therefore, Q-learning can learn from both off-policy (no control over actions) and on-policy (full control over actions) setting.
2. Optimistic Q-learning: $\hat{Q}_0^*(x,a) = \frac{R_{\max}}{1-\gamma}\prod_t (1-\alpha_t)^{-1}$ and pick $a_t = \arg\max_a \hat{Q}^*(x_t,a)$. Guaranteed that with prob. $1-\delta$, obtains ϵ-optimal in #steps polynomial in $|X|, |A|, 1/\epsilon, \log(1/\delta)$.

Pro: only store $\hat{Q}^*(x',a')$, which is $O(|X||A|)$.

Scaling up via Approximation

Idea: approximate tabular func by NN. Value approx: DQN; Policy approx: REINFORCE.

Note that tabular TD-learning can be viewed as SGD on $\ell_2(\theta; x, x', r) = (V(x;\theta) - r - \gamma V(x';\theta_{\text{old}}))^2$ or $(Q(x,a;\theta) - r - \gamma \max_{a'} Q(x',a';\theta_{old}))^2$. Therefore, we can use NN to approximate $V(x;\theta)$ or $Q(x,a;\theta)$. *DQN* also applies experience replay. To increase stability, *double DQN* use the current network to compute the argmax action (but still use the old network as the target).

Q-learning requires to compute the policy via $a_t = \textbf{argmax}\, Q(x_t,a;\theta)$. For large/continuous action space, this is intractable. REINFORCE use parametrized policy $\pi(x) := \pi(x;\theta)$ to avoid the argmax and optimize via policy gradient: maximize $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} r(\tau)$, where $r(\tau) = \sum \gamma^t r(x_t, a_t)$. The policy gradient is $\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)\nabla \log \pi_\theta(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)\sum_t \nabla \log \pi(a_t \mid x_t;\theta)] = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}(r(\tau) - b(\tau_{0:t-1}))\nabla \log \pi(a_t \mid x_t;\theta)]$ for any $b(\tau_{0:t-1})$. Let $G_t = \sum_{s=t}^{T} \gamma^{s-t} r^s$ to be the "reward to go" and $\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t G_t \nabla \log \pi(a_t \mid x_t;\theta)\right]$.

Advantage function: $A^\pi(x,a) = Q^\pi(x,a) - V^\pi(x)$, i.e., the advantage of picking a at current step instead of following π.

Actor-Critic

Idea: parametrize both the policy and the action-value function.

We can write the policy gradient (note reward-to-go is exactly $Q(s,a)$) as $\nabla J(\theta) = \mathbb{E}_{(x,a)\sim\pi_\theta}[Q(x,a;\theta_Q)\nabla \log \pi(a \mid x;\theta_\pi)]$. Actor-Crtic algorithm updates the parameters after observing (x,a,r,x'): $\theta_\pi \leftarrow \theta_\pi + \eta_t Q\left(x,a;\theta_Q\right)\nabla \log \pi(a \mid x;\theta_\pi)$ and $\theta_Q \leftarrow \theta_Q - \eta_t\left(Q\left(x,a;\theta_Q\right) - r - \gamma Q\left(x', \pi(x',\theta_\pi);\theta_Q\right)\right)\cdot \nabla Q(x,a;\theta_Q)$.

To generalize actor-critic to be off-policy, we can replace the $\max_a Q(x,a)$ by $Q(x,\pi(x))$, i.e., $L\left(\theta_Q\right) = \sum\left(r + \gamma Q\left(x', \pi(x';\theta_\pi);\theta_Q^{\text{old}}\right) - Q\left(x,a;\theta_Q\right)\right)^2$ and $\theta_\pi^* = \arg\max_\theta \mathbb{E}_{x\sim\mu}\left[Q\left(x,\pi(x;\theta);\theta_Q\right)\right]$.

6 Appendix

$\frac{\partial}{\partial\Sigma}\log|\Sigma| = \Sigma^{-T}$.