Lecture 8

Function Approximation and Policy Gradient Methods

Goals of this lecture

- Understand the motivation for value function approximation: generalization, scalability, and handling large/continuous state spaces.
- Learn how to define and minimize the value prediction error under a given policy.
- Introduce the notion of on-policy state distribution μ^π and how it shapes learning.
- Derive gradient descent and semi-gradient TD(0) updates for parametric approximators.
- Discuss convergence properties and tradeoffs in approximate value prediction.

8.1 Motivation for Function Approximation

In tabular reinforcement learning, we maintain a separate value estimate for each state (or state-action pair). While this approach is feasible for small discrete environments, it quickly breaks down as the state space grows. In high-dimensional or continuous environments, the number of states becomes too large to store or update individually, and generalization across similar states becomes essential.

To address this, we introduce *function approximation*, where we replace the tabular value function with a parameterized function:

$$\hat{v}(s;\theta) \approx v^\pi(s),$$

where $\hat{v} : \mathcal{S} \to \mathbb{R}$ is a differentiable function (e.g., linear model, neural network), and $\theta \in \mathbb{R}^d$ are the parameters to be learned.

Function approximation enables:

- **Scalability:** Storage and computation depend on the number of parameters d, not the number of states.
- **Generalization:** Updates to $\hat{v}(s;\theta)$ at one state can influence value estimates at similar states, improving learning efficiency.
- **Smoothness and Structure:** Prior knowledge (e.g., spatial locality, invariance) can be encoded through features or architectures.

This approach forms the foundation of modern deep reinforcement learning and allows RL agents to operate in large-scale and real-world domains.

8.2 Value Function Approximation

Given a parameterized value function $\hat{v}_\theta(s)$, our goal is to approximate the true state-value function $v^\pi(s)$ as accurately as possible. We measure accuracy via the *mean squared value error* (MSVE) under a suitable state distribution:

$$\mathcal{L}(\theta) := \mathbb{E}_{s\sim d^\pi}\left[(\hat{v}_\theta(s) - v^\pi(s))^2\right].$$

Stationary Distribution and Occupancy Measures. In reinforcement learning, we often measure the accuracy of value function approximation using expectations over a state distribution. Two commonly used distributions are the *stationary distribution* and the *discounted occupancy measure.*

Stationary Distribution. The *stationary distribution* $d^\pi(s)$ is defined as the limiting distribution over states when following a policy π in an ergodic Markov decision process.

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{P}(S_t = s|\pi, S_0 \sim \rho)$$

It satisfies the fixed-point equation:

$$d^\pi(s') = \sum_{s\in\mathcal{S}} d^\pi(s)\sum_{a\in\mathcal{A}}\pi(a \mid s)p(s' \mid s, a),$$

and reflects the long-run fraction of time spent in each state under policy π. Importantly, $d^\pi(s)$ is independent of the initial state distribution, provided the Markov chain is ergodic.

Discounted Occupancy Measure. In earlier lectures, we introduced the *discounted state occupancy measure*:

$$\rho_\pi^\gamma(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}(S_t = s \mid \pi, S_0 \sim \rho),$$

where ρ is an arbitrary initial state distribution. This quantity reflects the normalized expected number of discounted visits to state s when following policy π, starting from ρ.

Unlike the stationary distribution, the discounted occupancy measure *does* depend on the initial distribution. It becomes increasingly concentrated around frequently visited states as γ increases.

Limit Connection. Although ρ_π^γ and d^π are generally different, they coincide in the limit as $\gamma \to 1$ *only if* the initial distribution matches the stationary distribution:

$$\lim_{\gamma\to 1}\rho_\pi^\gamma(s) = d^\pi(s), \quad \text{if and only if} \quad \rho = d^\pi.$$

Otherwise, the limiting discounted distribution reflects a bias induced by the initial condition.

Practical Implication. When designing learning algorithms or analyzing performance, it is important to distinguish which distribution is used to define objectives. The stationary distribution d^π is natural when considering long-run average performance, while ρ_π^γ more accurately captures finite-horizon or discounted behavior starting from a given state distribution.

Trade-off and Approximation Accuracy. Function approximation inherently involves a trade-off. It is generally impossible to achieve perfect accuracy for every state, especially when state spaces are large or continuous. Therefore, the objective becomes minimizing error weighted by visitation frequencies of states under the policy π, thus prioritizing states that are frequently encountered.

Bootstrapped TD Targets. Since $v^\pi(s)$ is unknown, we use bootstrapped estimates from sampled experience. Given a sampled transition (S_t, R_{t+1}, S_{t+1}), the one-step Temporal-Difference (TD) target is:

$$\text{Target}_t = R_{t+1} + \gamma \hat{v}_\theta(S_{t+1}).$$

The stochastic approximation of MSVE then becomes:

$$\mathcal{L}_t(\theta) = \frac{1}{2}\left(\hat{v}_\theta(S_t) - [R_{t+1} + \gamma \hat{v}_\theta(S_{t+1})]\right)^2.$$

Stochastic Semi-gradient TD Update. Although this is not the exact gradient, the commonly used semi-gradient update rule is:

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta \hat{v}_\theta(S_t),$$

where the TD error δ_t is:

$$\delta_t = R_{t+1} + \gamma \hat{v}_\theta(S_{t+1}) - \hat{v}_\theta(S_t).$$

This update provides an efficient, incremental way to minimize the MSVE objective.

Linear TD(0) for Policy Evaluation A common practical choice is linear value function approximation:

$$\hat{v}_\theta(s) = \phi(s)^\top \theta,$$

with fixed feature vector $\phi(s) \in \mathbb{R}^d$ and parameter vector $\theta \in \mathbb{R}^d$.

The corresponding TD(0) update rule is:

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(S_t),$$

where the TD error becomes:

$$\delta_t = R_{t+1} + \gamma \phi(S_{t+1})^\top \theta_t - \phi(S_t)^\top \theta_t.$$

This update directly minimizes a projected Bellman error, ensuring convergence under standard assumptions. Rigorous analysis is provided by [1], employing stochastic approximation techniques and linear dynamical system theory.

Summary of Key Points:

- **MSVE** focuses approximation effort on states frequently visited under π.
- **Discounted occupancy measures** bridge initial distributions to stationary distributions.
- **Bootstrapping** via TD targets provides incremental and efficient updates.
- **Linear approximation** offers analytical tractability and guaranteed convergence.

8.3 From Value-Based to Policy-Based Methods

So far, we have considered *value-based* methods that estimate the value function v^π or q^π and indirectly improve the policy via greedy or ε-greedy updates. These methods rely heavily on estimating action values and can suffer from:

- High variance in long-horizon tasks when using Monte Carlo targets.
- Instability in function approximation due to moving targets and partial updates.
- Difficulty in representing or optimizing stochastic policies.
- Delayed or indirect credit assignment across time steps.

Direct Policy Optimization.

We now turn to *policy-based* methods that parameterize the policy $\pi_\theta(a \mid s)$ directly and optimize it to maximize long-term reward:

$$J(\theta) := \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} R_t\right].$$

This approach bypasses the need to explicitly estimate q^π and allows natural handling of continuous or stochastic policies.

Gradient Ascent. We seek to adjust θ via gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha\, \widehat{\nabla_\theta J(\theta)},$$

where $\widehat{\nabla_\theta J(\theta)}$ is a stochastic estimate of the policy gradient, computed from trajectories sampled from π_θ.

Advantages of Policy-Based Methods.

- Can naturally represent stochastic policies, which may be optimal in certain environments.
- Allow smooth optimization via gradient methods.
- Avoid maximization steps required by Q-learning or SARSA.

Policy Parameterization

To apply gradient-based optimization to policies, we define a differentiable, parameterized family of policies $\pi_\theta(a \mid s)$.

Discrete Action Spaces. A standard choice is the *softmax policy*:

$$\pi_\theta(a \mid s) = \frac{\exp(h(s, a; \theta))}{\sum_b \exp(h(s, b; \theta))},$$

where $h(s, a; \theta)$ is the **preference function**, expressing how favorable action a is in state s under parameters θ. A common choice is:

$$h(s, a; \theta) = \theta^\top x(s, a),$$

where $x(s, a) \in \mathbb{R}^d$ is a feature vector encoding the state-action pair.

Continuous Action Spaces. For continuous actions, a common parameterization is a Gaussian policy:

$$\pi_\theta(a \mid s) = \mathcal{N}(a \mid \mu_\theta(s), \sigma^2),$$

where the mean $\mu_\theta(s)$ is a function of the state and learned parameters θ, e.g., $\mu_\theta(s) = \theta^\top \phi(s)$. The variance σ^2 may be fixed or learned jointly.

Advantages.

- Softmax policies induce stochasticity that helps with exploration and robustness.
- Gaussian policies enable smooth action selection in continuous spaces.
- Both forms are differentiable, allowing for efficient gradient-based updates.

8.4 Policy Gradient Theorem and REINFORCE

Policy Gradient Theorem

To compute the gradient $\nabla_\theta J(\theta)$, we use the following result:

Theorem 8.1 (Policy Gradient Theorem)**.** *Let $\pi_\theta(\cdot \mid s)$ be a differentiable stochastic policy with parameter vector $\theta \in \mathbb{R}^d$, and let the performance objective be*

$$J(\theta) \;:=\; \mathbb{E}_{\pi_\theta}\Big[\textstyle\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t)\Big], \quad 0 \le \gamma < 1.$$

Then

$$\nabla_\theta J(\theta) \;=\; \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} q^\pi(s,a)\, \nabla_\theta \pi_\theta(a \mid s),$$

where $d^\pi(s) = \lim_{t\to\infty} \mathbb{P}\big(S_t = s \;\big|\; S_0 \sim \rho, \pi\big)$ is the stationary distribution under π_θ (assumed to exist and be unique) and $q^\pi(s,a) = \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k\, r(S_{t+k}, A_{t+k}) \mid S_t = s, A_t = a]$ is the action–value function.

Log-Derivative Trick. Since the gradient of a probability is tricky to work with, we reparametrize:

$$\nabla_\theta \pi_\theta(a \mid s) = \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s),$$

and rewrite the gradient as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[q^\pi(s,a) \nabla_\theta \log \pi_\theta(a \mid s)\right].$$

Proof. **Step 1: Gradient of v^π.** For every state s, $v^\pi(s) = \sum_a \pi_\theta(a \mid s) q^\pi(s,a)$, so by the product rule

$$\nabla_\theta v^\pi(s) = \sum_a \big[\nabla_\theta \pi_\theta(a \mid s)\big]\, q^\pi(s,a) + \sum_a \pi_\theta(a \mid s)\, \nabla_\theta q^\pi(s,a). \tag{8.1}$$

Step 2: Recursive expansion of $\nabla_\theta q^\pi$. Using the one-step Bellman equation $q^\pi(s,a) = \sum_{s',r} p(s', r \mid s,a)[\,r + \gamma v^\pi(s')]$ and differentiating,

$$\nabla_\theta q^\pi(s,a) = \gamma \sum_{s'} p(s' \mid s,a)\, \nabla_\theta v^\pi(s').$$

Step 3: Substitute into (8.1). Plugging the above into (8.1) gives

$$\nabla_\theta v^\pi(s) = \sum_a \nabla_\theta \pi_\theta(a \mid s)\, q^\pi(s,a) + \gamma \sum_a \pi_\theta(a \mid s) \sum_{s'} p(s' \mid s,a)\, \nabla_\theta v^\pi(s').$$

Define the *transition matrix under* π_θ as $P_\pi(s,s') := \sum_a \pi_\theta(a \mid s) p(s' \mid s,a)$. In vector form:

$$\nabla_\theta v^\pi = \underbrace{\Phi}_{\text{size } |\mathcal{S}| \times d} + \gamma P_\pi\, \nabla_\theta v^\pi, \quad \text{where } \Phi(s) := \sum_a \nabla_\theta \pi_\theta(a \mid s)\, q^\pi(s,a).$$

Solve the linear system: $\nabla_\theta v^\pi = (I - \gamma P_\pi)^{-1} \Phi$.
Step 4: Gradient of $J(\theta)$. Because $J(\theta) = \sum_s d^\pi(s) v^\pi(s)$ (with d^π the stationary distribution),

$$\nabla_\theta J(\theta) = \sum_s d^\pi(s)\, \nabla_\theta v^\pi(s) = \sum_s \left[d^{\pi\top} (I - \gamma P_\pi)^{-1} \right]_s \Phi(s).$$

For $\gamma < 1$ and ergodic π_θ, $d^{\pi\top}(I - \gamma P_\pi)^{-1} = (1-\gamma) \sum_{t=0}^\infty \gamma^t d^{\pi\top} P_\pi^t = d^\pi(\cdot)^\top$, the discounted–occupancy vector, which equals d^π. Hence

$$\nabla_\theta J(\theta) = \sum_s d^\pi(s) \sum_a \nabla_\theta \pi_\theta(a \mid s)\, q^\pi(s,a),$$

which is exactly the claimed result. $\square$

REINFORCE: Monte Carlo Policy Gradient

REINFORCE is a Monte Carlo algorithm that applies the policy gradient theorem using full returns:

$$G_t := \sum_{k=t}^{T} \gamma^{k-t} R_k.$$

It replaces $q^\pi(s,a)$ with a sampled return G_t, yielding the update:

$$\theta \leftarrow \theta + \alpha\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t.$$

Interpretation. REINFORCE performs stochastic gradient ascent on $J(\theta)$ using unbiased estimates of the gradient. However, it suffers from high variance due to long returns G_t, motivating the use of variance reduction techniques, such as baselines, which we will discuss next.

8.5 Variance Reduction via Baselines

The policy gradient estimator used in REINFORCE is unbiased, but often exhibits high variance due to the use of complete returns G_t. A powerful variance reduction technique is to subtract a *baseline* $b(s_t)$ from the return without introducing bias:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, (G_t - b(s_t)) \right].$$

```
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic)
Input: differentiable policy parameterization $\pi_\theta(a \mid s)$
Parameter: step size $\alpha > 0$
Initialize: policy parameters $\theta \in \mathbb{R}^d$ (e.g. $\theta = 0$)
Loop forever (for each episode)
  1. Generate an episode $\{S_0, A_0, R_1, \ldots, S_T, A_T, R_{T+1}\}$ following $\pi_\theta$.
  2. For each step $t = 0, 1, \ldots, T$ do
     (a) Compute the return                                              ($G_t$)
             $G_t = \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}$.
     (b) Update the policy parameters                       (semi-gradient step)
             $\theta \leftarrow \theta + \alpha\, \gamma^t\, G_t\, \nabla_\theta \ln \pi_\theta(A_t \mid S_t)$.
```

Figure 8.1: The REINFORCE algorithm uses complete returns to form an unbiased policy-gradient estimate and updates parameters after every episode.

Choice of Baseline and Variance Reduction. In REINFORCE and other Monte Carlo policy gradient methods, we estimate the gradient of the performance objective by sampling full returns. One key insight is that we can reduce the variance of this estimator—without introducing bias—by subtracting a *baseline* from the return:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(A_t \mid S_t)\,(G_t - b(S_t))\right],$$

for any function $b : \mathcal{S} \to \mathbb{R}$.

This identity holds because the expected value of the baseline term vanishes:

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_t \mid S_t) b(S_t)\right] = \mathbb{E}_{\pi_\theta}\left[b(S_t) \cdot \underbrace{\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(A_t \mid S_t) \mid S_t]}_{=0}\right] = 0.$$

Optimal Baseline. The variance of the policy gradient estimator is minimized when the baseline is chosen as the expected return conditioned on the current state:

$$b^\star(s) = \mathbb{E}_{\pi_\theta}[G_t \mid S_t = s] = v^{\pi_\theta}(s).$$

That is, the **state-value function** is the variance-optimal baseline..

Actor–Critic Architecture. In practice, $v^{\pi_\theta}(s)$ is not known and must be estimated. This leads to the actor–critic framework:

- The **critic** learns an estimate $\hat{v}_\omega(s) \approx v^{\pi_\theta}(s)$ using value function approximation (e.g., TD learning),
- The **actor** uses this estimate to compute the advantage

  $$A_t := G_t - \hat{v}_\omega(S_t),$$

  and updates θ using:

  $$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(A_t \mid S_t) A_t.$$

This reduces the variance of the gradient estimator without changing its expected value, improving the stability and sample efficiency of learning.

8.6 Summary and Discussion

- **Function approximation** enables reinforcement learning in large or continuous spaces, trading off expressiveness and approximation error.
- **TD methods** remain effective and stable under linear approximations, and are widely used in practice (e.g., TD(0), SARSA).
- **Policy gradients** allow direct optimization of stochastic policies, offering flexibility and robustness, especially in high-dimensional or continuous action spaces.
- **REINFORCE** provides a conceptually simple algorithm for Monte Carlo policy gradient estimation, but suffers from high variance.
- **Baselines** reduce variance in policy gradient methods without introducing bias; actor-critic methods leverage this by learning the baseline from data.

These ideas form the foundation of modern reinforcement learning algorithms, particularly in settings involving neural networks, continuous control, and stochastic environments.

References for Lecture 8

[1] John Tsitsiklis and Benjamin Van Roy. "Analysis of temporal-diffference learning with function approximation". In: *Advances in neural information processing systems* 9 (1996).