Value Function Approximation

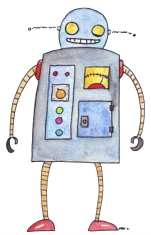
(INF8250AE: Introduction to Reinforcement Learning)

each credit : 45h

Amir-massoud Farahmand

Polytechnique Montréal & Mila

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE



Adaptive Agents Lab (Adage)

Mila

Table of Contents

1. Motivation and Function Approximation
2. Overview of Supervised Learning and Loss Functions
3. Value Function Computation with a Function Approximator
   - Approximation Given the Value Function
   - Approximate Value Iteration (Population)
   - Bellman Residual Minimization (Population Version)
   - Projected Bellman Error (Population Version)
     - Least Squares Temporal Difference Learning (Population)
     - Fixed Point Iteration for Projected Bellman Operator
     - Error Bound on the LSTD Solution
4. Batch RL Methods
   - Value Function Approximation Given the Monte Carlo Estimates
   - Approximate Value Iteration
   - Bellman Residual Minimization
   - LSTD and Least Squares Policy Iteration (LSPI)
5. Online RL Methods
   - Semi-Gradient Viewpoint

Goal

We study value function learning under function approximation.

- Several loss functions to learn the value function
- Batch (Offline) RL
- Online RL

Learning Objectives

You need to

- Remember: What value function approximation is; Approximate Value Iteration; Bellman Error; Projected Bellman Error; LSTD; Fitted Value Iteration; TD Learning with VFA
- Understand: How to define population loss functions for value function learning; How to estimate those losses using samples; Online methods
- Apply: Learning algorithms to estimate value functions

Motivation

In many real-world problems, the state-action space $\mathcal{X} \times \mathcal{A}$ is so large that we cannot represent quantities such as the value function or policy **exactly**.

- $\mathcal{X} \subset \mathbb{R}^d$ with $d \geq 1$. Exact representation of an arbitrary function on $\mathbb{R}^d$, or even on $\mathbb{R}$, on a computer is infeasible.
- $\mathcal{X}$ is finite, but very large (millions or billions).

We need to approximate those functions using a representation that is feasible to manipulate on a computer. This is called function approximation (FA).
Function approximation is important for generalization too.

Function Approximation in RL

Function approximation is used and studied in many fields: approximation theory, ML/Statistics.
In RL, it shows up in different ways:

- Value function approximation: $\hat{V}^\pi \approx V^\pi$ or $\hat{V}^* \approx V^*$
- Policy approximation: $\hat{\pi}^* \approx \pi^*$.
- Model approximation: $\hat{\mathcal{P}} \approx \mathcal{P}$ and $\hat{r} \approx r$.

These function approximators should be easily represented on a computer.

- This lecture: Value function approximation.
- Next lecture: Policy approximation

Linear Function Approximation

We may use a linear function approximator defined based on a set of basis functions (or features), i.e.,

$$\hat{V}(x) = \phi(x)^\top w = \sum_{i=1}^{p} \phi_i(x) w_i,$$

with $w \in \mathbb{R}^p$ and $\phi : \mathcal{X} \to \mathbb{R}^p$.
Any $\hat{V}$ belongs to the space of functions $\mathcal{F}$

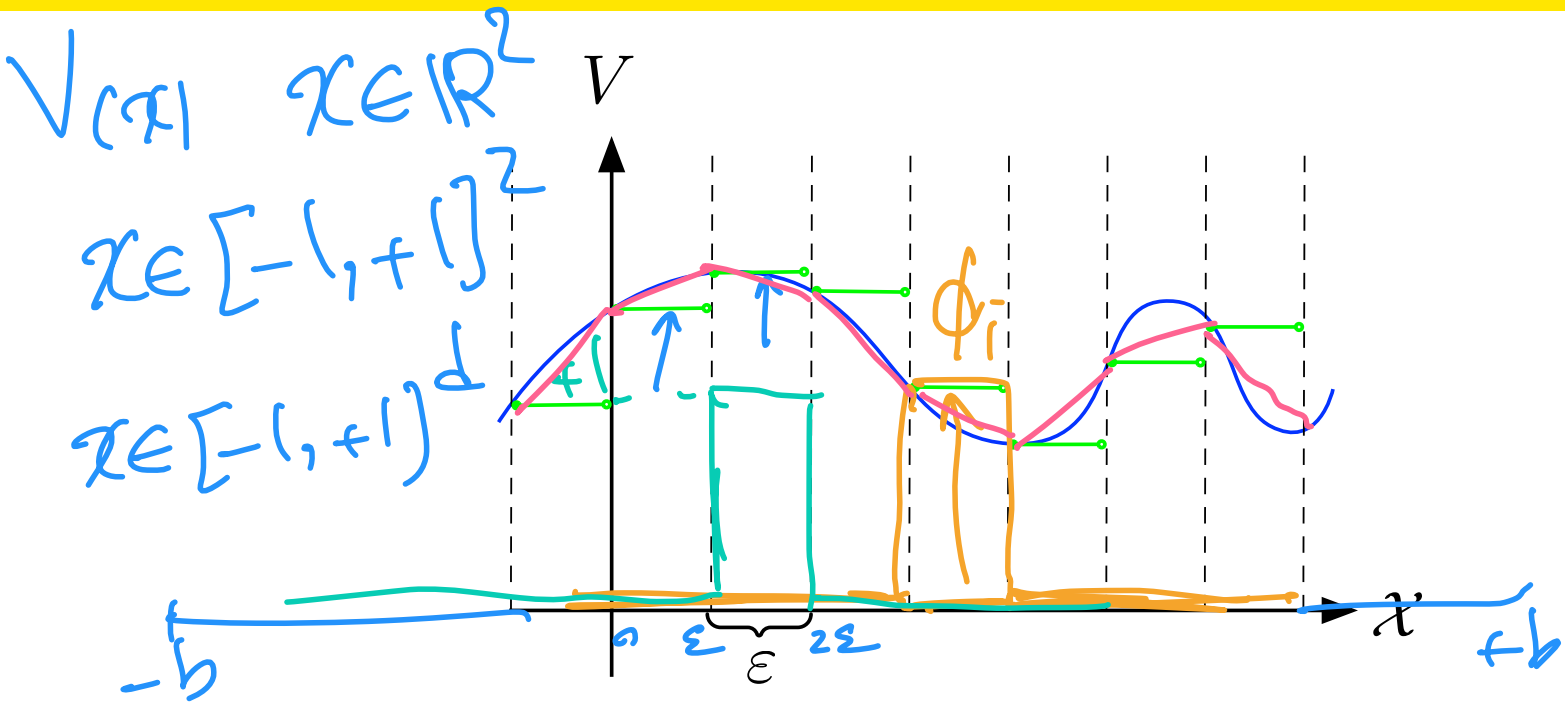
$$\mathcal{F} = \left\{ x \mapsto \phi(x)^\top w \ : \ w \in \mathbb{R}^p \right\}. \tag{1}$$

The function space $\mathcal{F}$ is called the value function space. In this example, it is a span of a set of features. We simply call it a linear function space.

Remark

The linearity is in the parameters w and not in the state x. For example, the function $\hat{V}(x) = \sin(2\pi x) w_1 + \cos(2\pi x) w_2$ is a linear function in weights, but $\hat{V}(x) = x w_1 w_2$ is not linear function in the weights.

Piecewise Constant Approximation



Assume that the domain is $[-b, +b]$, we can define ϕ_i (for $i = 0, 1, \ldots, \lceil \frac{2b}{\varepsilon} - 1 \rceil$) as

$$\phi_i(x) = \mathbb{I}\{x \in [-b + i\varepsilon, -b + (i+1)\varepsilon)\}.$$

Any function V can be approximated by a $\hat{V}(x) = \hat{V}(x; w) = \phi(x)^\top w$ with $w \in \mathbb{R}^{\lceil \frac{2b}{\varepsilon} \rceil}$. So it is a linear function approximator. Denote such a function space by $\mathcal{F}_\varepsilon$.

Piecewise Constant Approximation

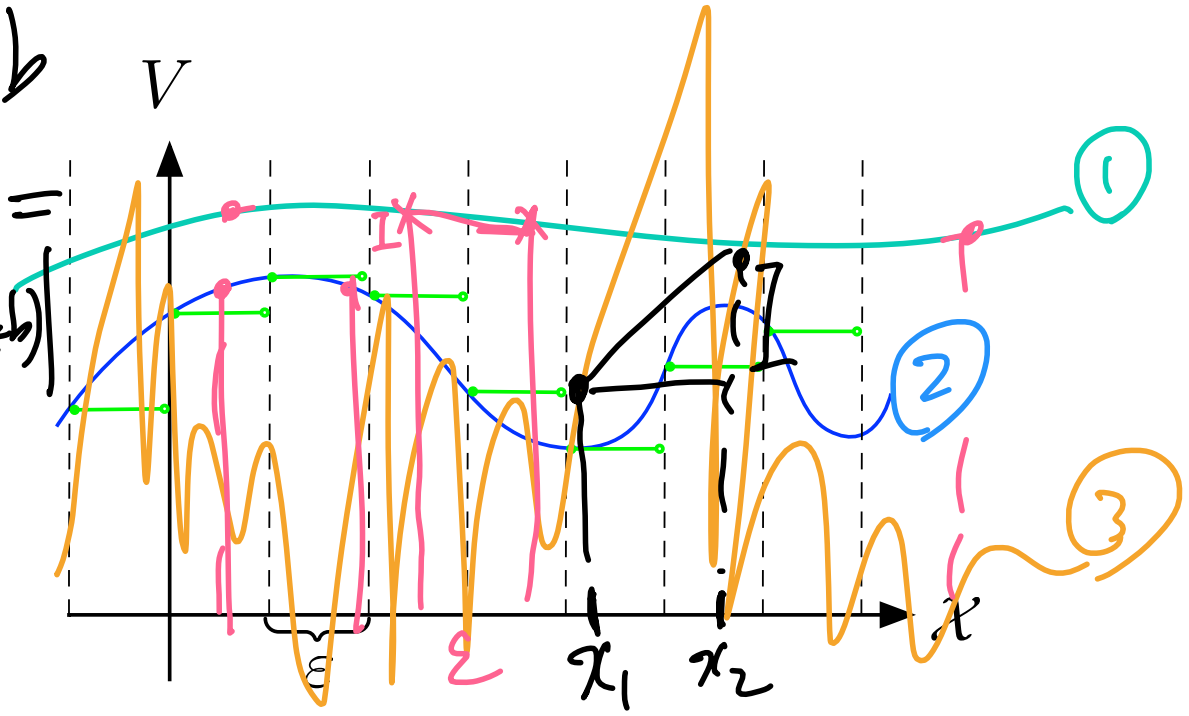
$V(x) = ax + b$

$|V(x_1) - V(x_2)| =$

$|(ax_1 + b) - (ax_2 + b)|$

$= |a(x_1 - x_2)|$

$= |a| \cdot |x_1 - x_2|$



Questions:

- How accurately can we approximate function V?
- How many parameters do we require in order to represent this function approximator?
- What is the effect of the function approximation on statistical estimation?

Approximation Error

- The approximation quality depends on the regularity or structure of the value function V.
- If we allow V to change arbitrarily, we cannot hope to have a good approximation.

Approximation Error

$|V(x_1) - V(x_2)| \leq L \cdot \|x_1 - x_2\| \quad \forall x_1, x_2$

- If the value function has some regularities, we can say more.
- For any V that is L-Lipschitz,

$$\inf_{\hat{V} \in \mathcal{F}_\varepsilon} \left\| \hat{V} - V \right\|_\infty \leq L\varepsilon.$$

ε ? ε^2

- This is called the approximation error or bias.
- Approximation error depends on:
  - the structure of the function approximator, e.g., piecewise constant, piecewise linear, etc.
  - the class of functions that is being approximated, e.g., L-Lipschitz functions.

Curse of Dimensionality in the Number of Parameters

If the domain was $\mathcal{X} = [-1, +1]^d$ for $d \geq 1$, we would need

$$O\left(\frac{1}{\varepsilon^d}\right)$$

parameters to describe such a function.
This increases exponentially fast as a function of d.
Example: With $\varepsilon = 0.1$ and $d = 2$, we need $(\frac{2}{0.1})^2 = 400$ parameters; with $d = 10$, we need $2^{10} \times 10^{10}$ parameters.
This exponential growth of the number of parameters required to represent a high-dimensional function is an instance of the curse of dimensionality.

Estimation Error

- We also need to pay attention to the statistical aspect of estimating a function within this function space using a finite number of data points.
- The estimation accuracy depends on some notion of complexity or size of $\mathcal{F}$.
- Many ways to quantify the complexity of $\mathcal{F}$.
  - VC Dimension
  - Covering Number / Metric Entropy
  - Rademacher Complexity

Estimation Error

- Quantifying the complexity of function space $\mathcal{F}$ requires some further development. Briefly speaking, the statistical error behaves as

$$O(\sqrt{\frac{\log |\mathcal{F}|}{n}}),$$

where n is the number of data points used in the estimation.

  - Recall that the variance of Sample Average Estimator m_t with t data point is $\frac{\sigma^2}{t}$. This means that $\mathbb{E}\left[|m_t - \mu|^2\right] = \mathbb{E}\left[|m_t - \mathbb{E}\left[m_t\right]|^2\right] = \frac{\sigma^2}{t}$. This implies that $\mathbb{E}\left[|m_t - \mu|\right] \leq \sqrt{\mathbb{E}\left[|m_t - \mu|^2\right]} = \frac{\sigma}{\sqrt{t}}$.
  - Compare with $O(\sqrt{\frac{\log |\mathcal{F}|}{n}})$ here. For the mean estimation problem, $|\mathcal{F}|$ is effectively equal to 1 (the logarithmic term $\log |\mathcal{F}|$ in the upper bound is actually $\log c|\mathcal{F}|$ for a constant c, so we are not facing $\log 1 = 0$ situation; that constant is absorbed in the O notation).

- This is called the estimation error or variance.

Different Function Approximators

There are many other ways to represent the value function approximation $\hat{V}$ (and effectively, $\mathcal{F}$). Some examples are

- Deep neural networks (DNN)
- reproducing kernel Hilbert spaces (RKHS)
- Decision trees and random forests
- Local methods such as smoothing kernels, k-nearest neighbours, etc.

Overview of Supervised Learning and Loss Functions

Supervised Learning Problem

Suppose that there is a relationship between an input variable $x \in \mathcal{X}$ and output variable $Y \in \mathcal{Y}$.
The relation may be:

- Deterministic: $y = f^*(x)$
- Stochastic: $Y \sim P(\cdot|X)$

In either case, $f^* : \mathcal{X} \to \mathbb{R}$ or $P(\cdot|x)$ are unknown.
For simplicity, assume that $Y \subset \mathbb{R}$, so $f^* : \mathcal{X} \to \mathbb{R}$.
We can think of the stochastic one as $Y(x) = f^*(x) + \varepsilon(x)$, with $f^*(x) = \mathbb{E}[Y|X = x]$ the conditional expectation of the output Y and the conditionally zero-mean noise ε being added to it ($\mathbb{E}[\varepsilon|X = x] = 0$).
When we do not know $f^* : \mathcal{X} \to \mathbb{R}$ or $P(\cdot|x)$, we can try to learn them based on data.

Supervised Learning Problem

In the supervised learning problem, we are given a dataset

$$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n,$$

with $X \sim \mu$ and $Y \sim P(\cdot|X)$ (or $Y(x) = f^*(x) + \varepsilon$).
Q: How can we use the dataset $\mathcal{D}_n$ to find a function $\hat{f}$ that is approximately close to f^*?
First, we need a way to measure how close $\hat{f}$ is to f^*.
A common way is to define a loss (or risk) function.

Pointwise Loss Function

Let us start by focusing on measuring the difference between $\hat{f}$ and f^* at a single point. This is measured by the pointwise loss function.
A particular choice of pointwise loss is the squared error loss:
If instead of y_{true}, our predictor f predicts $\hat{y}$, the squared error is

$$l(\hat{y}, y_{\text{true}}) = |\hat{y} - y_{\text{true}}|^2.$$

We overload the notation and write the loss for a predictor $f : \mathcal{X} \to \mathbb{R}$ as

$$l(x, y) = l(x, y; f) = l(f(x), y) = |f(x) - y|^2.$$

So for an input data point x and its corresponding (random) target $Y(x) \sim P(\cdot|x)$ (or $Y(x) = f^*(x) + \varepsilon$), this loss for a predictor f is

$$l(x, Y(x)) = |f(x) - Y(x)|^2.$$

Pointwise Loss Function

Consider the pointwise loss:

$$l(x, Y(x)) = |f(x) - Y(x)|^2.$$

When is this loss small?
Let us expand it to understand this term better:

$$\begin{aligned} \mathbb{E}\left[l(x, Y(x))\right] &= |f(x) - Y(x)|^2 \\ &= |f(x) - f^*(x) - \varepsilon|^2 \\ &= |f(x) - f^*(x)|^2 + |\varepsilon|^2 - 2\left(f(x) - f^*(x)\right)\varepsilon. \end{aligned}$$

- The first term depends on the difference between $f(x)$ and $f^*(x)$.
- The second term only depends on the noise, so it is not dependent on f.
- The third term is an inner product of the noise and the residual $f(x) - f^*(x)$. Let us study it closely.

Pointwise Loss Function

We take the expectation conditioned on a fixed x of the pointwise loss. This means that we only consider the randomness of Y and let x be fixed:

$$\begin{aligned}\mathbb{E}\left[l(x, Y(x))|X=x\right] =& \mathbb{E}\left[\left|f(x)-f^*(x)\right|^2|X=x\right]+\mathbb{E}\left[|\varepsilon|^2|X=x\right] \\ & -2\mathbb{E}\left[\left(f(x)-f^*(x)\right)\varepsilon|X=x\right] \\ =& \left|f(x)-f^*(x)\right|^2+\mathbb{E}\left[\varepsilon^2|X=x\right] \\ & -2\left(f(x)-f^*(x)\right)\underbrace{\mathbb{E}\left[\varepsilon|X=x\right]}_{=0} \\ =& \left|f(x)-f^*(x)\right|^2+\operatorname{Var}\left[\varepsilon|X=x\right].\end{aligned}$$

Q: When is this minimized?

Hint: Note that $\operatorname{argmin}_z\{g(z)+c\}=\operatorname{argmin}_z\{g(z)\}$.

Pointwise Loss Functions

When is this conditional expectation minimized?

$$\hat{f} \leftarrow \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \mathbb{E}\left[l(x, Y(x))|X = x\right]$$

As

$$\mathbb{E}\left[l(x, Y(x))|X = x\right] = \left|f(x) - f^*(x)\right|^2 + \operatorname{Var}\left[\varepsilon|X = x\right],$$

Whatever we f choose, it does not affect the variance term $\operatorname{Var}\left[\varepsilon|X = x\right]$.
If there is no restriction in the choice of f (meaning that $\mathcal{F}$ is very flexible), the first term is minimized when $f(x) = f^*(x)$.
So the minimizer at point x is a function $\hat{f}$ such that its value at x is $\hat{f}(x) = f^*(x)$. Its value at other points in $\mathcal{X}$ can be anything.
What we care about is to find an $\hat{f}$ that is close to f^* for all $x \in \mathcal{X}$, not just a single x.
Can we define a loss for the whole input space $\mathcal{X}$?

Population Loss Function

We can extend the definition of the loss from a single x to the whole input space.

$$L(f) \triangleq \mathbb{E}\left[|f(X) - Y|^2\right], \tag{2}$$

where $X \sim \mu$ and $Y \sim P(\cdot|X)$. The distribution μ gives potentially different weight to different inputs $x \in \mathcal{X}$. This is called population loss (or risk) function.
We have that

$$\begin{aligned} L(f) &= \mathbb{E}\left[|f(X) - Y|^2\right] = \mathbb{E}\left[\mathbb{E}\left[l(X, Y(X))|X\right]\right] \\ &= \mathbb{E}\left[|f(X) - f^*(X)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[\varepsilon|X\right]\right]. \end{aligned}$$

Population Loss Function

$$L(f) = \mathbb{E}\left[|f(X) - Y|^2\right] = \mathbb{E}\left[|f(X) - f^*(X)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[\varepsilon|X\right]\right].$$

The second term ($\mathbb{E}\left[\mathrm{Var}\left[\varepsilon|X\right]\right]$) does not depend on the predictor f. So the population loss is minimized whenever the first term $\mathbb{E}\left[|f(X) - f^*(X)|^2\right]$ is minimized. This means that

$$\operatorname*{argmin}_{f\in\mathcal{F}} \mathbb{E}\left[|f(X) - Y|^2\right] = \operatorname*{argmin}_{f\in\mathcal{F}} \mathbb{E}\left[|f(X) - f^*(X)|^2\right] \tag{3}$$

If there is no restriction on the choice of f, the minimizer is $f(x) = f^*(x)$ for all x on the support of μ (recall: $\mathrm{supp}(\mu) = \{\, x : \mu(\mathrm{d}x) > 0 \,\}$). (Q: Why?!)

Note: More precisely, $f = f^*$ μ-almost surely.

Population Loss Function

We can try to find the minimizer $\hat{f}$ within a function space $\mathcal{F}$:

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \mathbb{E}\left[|f(X) - Y|^2\right].$$

How well does this procedure work?

- If $\mathcal{F}$ is a flexible enough family of functions, we can find an $\hat{f}$ that minimizes $\mathbb{E}\left[|f(X) - f^*(X)|^2\right]$ quite well.
- In the best possible scenario, $\hat{f} = f^*$ for all $\text{supp}(\mu)$.
- If $\mathcal{F}$ is not a rich enough class of functions, we cannot find a good approximation. This is the approximation error we encountered before.

But there is a catch: We cannot solve this problem because we often cannot compute the expectation; we are only given data points $\mathcal{D}_n$, and not the whole probability distribution.
So instead, we define another loss function using data.

Empirical Loss Functions

Given a dataset $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$, we define the empirical loss (or risk) of function f as

$$L_n(f) \triangleq \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2.$$

We then minimize

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2. + \lambda J(f)$$

This procedure is called Empirical Risk Minimization (ERM).
This is the basis of many ML algorithms.

Value Function Computation with a Function Approximator

Value Function Computation with a Function Approximator

- Let us develop some general approaches for the value function computation when we are restricted to use functions from a value function space $\mathcal{F}$.
- Most of the approaches are based on defining a loss function that should be minimized in order to find an approximate value function.
  - We define several loss functions, some of which benefit from the recursive structure of the MDP.
- The presentation in this section focuses on the population version of these approaches, when we have access to the model. We extend them to the sample-based variants in Section 3.

Reminder: L_p-Norms

For a probability distribution $\nu \in \mathcal{M}(\mathcal{X})$, and a function $V \in \mathcal{F}$, we define the $L_p(\nu)$-norm of V with $1 \leq p < \infty$ as

$$\|V\|_{p,\nu}^p \triangleq \int_{\mathcal{X}} |V(x)|^p d\nu(x). \tag{4}$$

The $L_\infty(\mathcal{X})$-norm is

$$\|V\|_\infty \triangleq \sup_{x \in \mathcal{X}} |V(x)|.$$

If we want to emphasize that the probability distribution is defined on the state space $\mathcal{X}$, we use $\nu_{\mathcal{X}}$ and $\|V\|_{p,\nu_{\mathcal{X}}}$.

$\|Q\|_{p,\nu}$

Approximation Given the Value Function V^π



Approximation Given the Value Function

Suppose that we happen to know V^π (or Q^π, V^*, Q^*), and we want to represent it with a function $V \in \mathcal{F}$.

Goal: finding $V \in \mathcal{F}$ such that

$$V \approx V^\pi.$$

Approximation Given the Value Function

To quantify $V \approx V^\pi$, we have to pick a distance function between function V and V^π, i.e., $d : \mathcal{B}(\mathcal{X}) \times \mathcal{B}(\mathcal{X}) \to \mathbb{R}$.
Given such a distance function, we can express our goal as

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} d(V, V^\pi).$$

Approximation Given the Value Function

We can use the L_p-norm w.r.t. a (probability) distribution $\mu \in \mathcal{M}(\mathcal{X})$ to define distances between functions.
We define the distance between functions V_1 and V_2 based on the norm as

$$d(V_1, V_2) = \|V_1 - V_2\|_{p,\mu} .$$

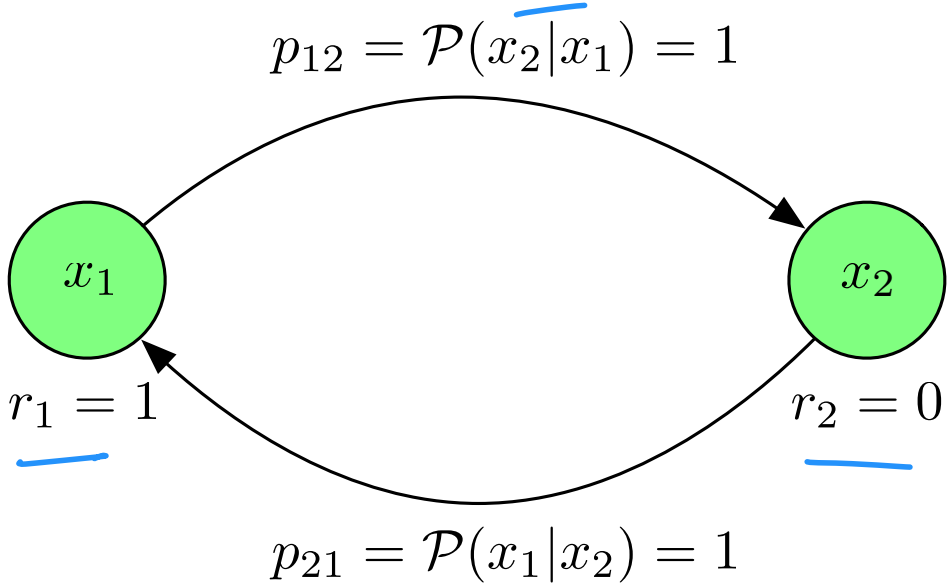
We can then have

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - V^\pi\|_{p,\mu} . \tag{5}$$

A common choice is the L_2-norm.
This should remind us of the mean squared loss function commonly used in regression, and we faced earlier as (2) in Section 2.

Example: Bases Matter

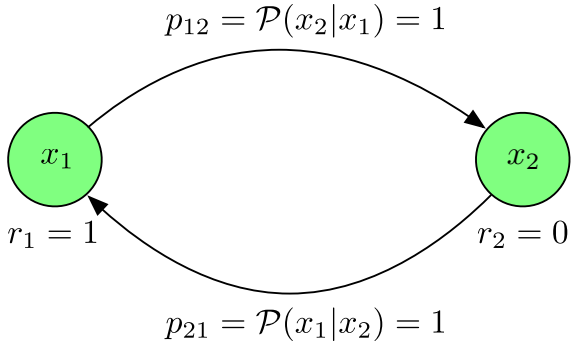
Consider the following MDP with $\gamma = \frac{1}{2}$.



The exact value function can be computed by solving the Bellman Equation:

$$\left.\begin{aligned} V^\pi(x_1) &= 1 + \frac{1}{2}V^\pi(x_2) = 1 + \frac{1}{4}V^\pi(x_1) \\ V^\pi(x_2) &= \frac{1}{2}V^\pi(x_1) \end{aligned}\right\} \Rightarrow V^\pi = \begin{bmatrix} \frac{4}{3} \\ \frac{2}{3} \end{bmatrix}$$

Example: Bases Matter



$$V^\pi = \begin{bmatrix} \frac{4}{3} \\ \frac{2}{3} \end{bmatrix}$$

Suppose we would like to use a function approximator with a single basis function $\phi = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Here the parameter describing the value function approximator is $w \in \mathbb{R}$ and

$$\hat{V}(w) = \phi w = \begin{bmatrix} w \\ 2w \end{bmatrix}.$$

Q: Is there any w that can exactly represents V^π?

Example: Bases Matter

$$V^\pi = \begin{bmatrix} \frac{4}{3} \\ \frac{2}{3} \end{bmatrix} \approx \begin{bmatrix} 1.333 \\ 0.666 \end{bmatrix}$$

Given the value function approximator

$$\hat{V}(w) = \phi w = \begin{bmatrix} w \\ 2w \end{bmatrix}.$$

what is the best w we can find to approximate V^π?
Suppose we use equal weighting on both states (μ is uniform), we get that

$$\min_w \left\| \hat{V}(w) - V^\pi \right\|_2^2 = \left(w - \frac{4}{3} \right)^2 + \left(2w - \frac{2}{3} \right)^2$$

$$\Rightarrow w^* = \frac{8}{15} \quad \text{and} \quad \hat{V}(w^*) = \begin{bmatrix} \frac{8}{15} \\ \frac{16}{15} \end{bmatrix} \approx \begin{bmatrix} 0.53 \\ 1.06 \end{bmatrix}.$$

Example: Bases Matter

$$V^\pi = \begin{bmatrix} \frac{4}{3} \\ \frac{2}{3} \end{bmatrix} \approx \begin{bmatrix} 1.333 \\ 0.666 \end{bmatrix}$$

Now suppose we use a different basis function $\phi = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. As before, the parameter describing the value function approximator is $w \in \mathbb{R}$ and

$$\hat{V}(w) = \phi w = \begin{bmatrix} 2w \\ w \end{bmatrix}.$$

Given the value function approximator

$$\hat{V}(w) = \phi w = \begin{bmatrix} 2w \\ w \end{bmatrix}.$$

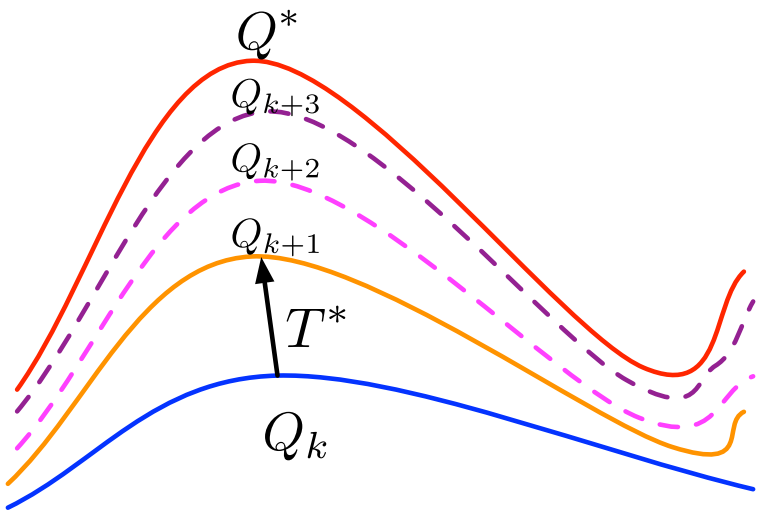
what is the best w we can find to approximate V^π?

Approximation Given the Value Function: How to Get V^π?

- How can we even have access to V^π?
  - The Monte Carlo (MC) estimate: For a state x, we can have $V^\pi(x) + \varepsilon(x)$ with $\mathbb{E}[\varepsilon(x)] = 0$.
- If MC can give an estimate for the value of a state, what is the reason for using function approximation after all?
  - When the state space is large (e.g., continuous), we cannot run MC for all states, but only for a finite number of them.
  - MC estimate is often noisy.
- The role of FA is to help us generalize from a finite number of noisy data points to the whole state space.
- MC estimate does not benefit from the structural properties of MDP.

Approximate Value Iteration

Approximate Value Iteration (Population Version)



Recall that VI is

$$V_{k+1} \leftarrow TV_k,$$

with T being either T^π or T^*.

One way to develop its approximate version is to perform each step only approximately, i.e., find $V_{k+1} \in \mathcal{F}$ such that

$$V_{k+1} \approx TV_k.$$

Approximate Value Iteration (Population Version)

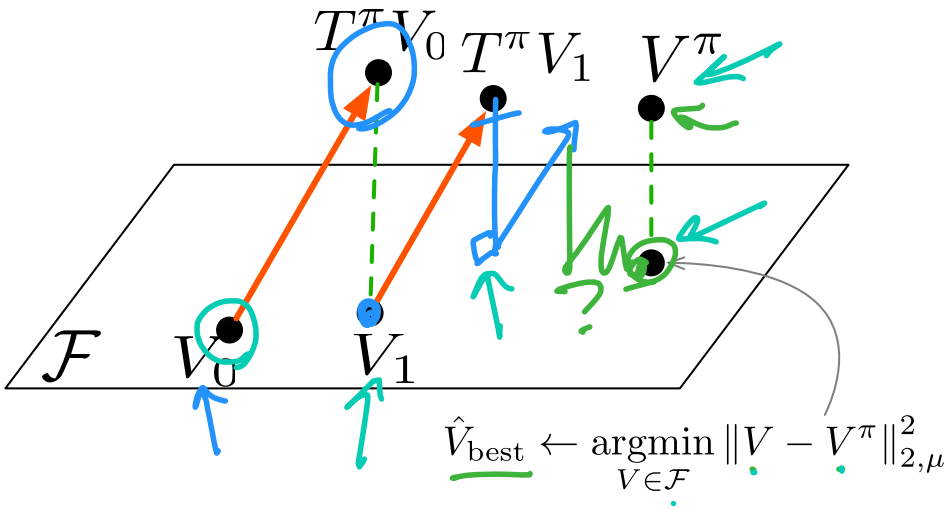
$V_k \in \mathcal{F} \longrightarrow TV_{k+1} \in \mathcal{F}$?

We start from a $V_0 \in \mathcal{F}$, and then at each iteration k of AVI we solve

$$V_{k+1} \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - TV_k\|_{p,\mu}. \tag{6}$$

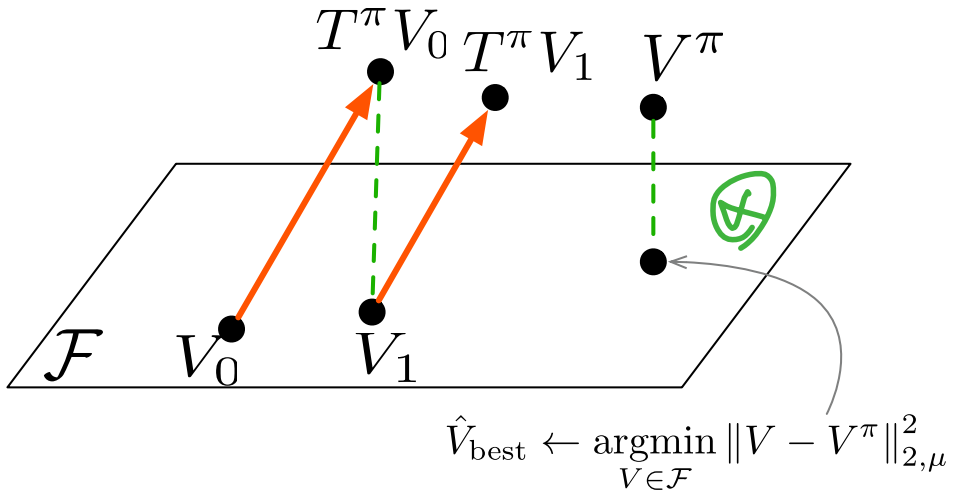
The procedure for the action-value function is similar.

Approximate Value Iteration (Population Version): A Geometric View



- When $\mathcal{F}$ is the set of linear functions (1), its geometry is the subspace spanned by ϕ.
- Even though $V_k \in \mathcal{F}$, the function TV_k may not be within $\mathcal{F}$ anymore ($TV_k \notin \mathcal{F}$). We visualize it with a point outside the plane.

Approximate Value Iteration (Population Version)



- The amount of this error depends on how expressive $\mathcal{F}$ is and how much T can push a function within $\mathcal{F}$ outside that space.
- Although VI is a convergent procedure (recall: T is γ-contraction), AVI may sometimes not converge, and it may indeed diverge.

Bellman Residual Minimization

Bellman Residual Minimization (Population Version)

- AVI tries to benefit from the contraction property of the Bellman operator, which allowed VI to be a sound solution. As mentioned, this may not always work under FA.
- Let us benefit from another structural property of an MDP.
- If we find a V such that $V = T^\pi V$, that function must be V^π (recall: The Bellman equation has a unique solution).
- Under FA, we may not achieve this exact equality.
- Instead we try to find a $V \in \mathcal{F}$ that approximately satisfies the equality:

$$V \approx T^\pi V.$$

- We can think of different ways to quantify the quality of approximation.
- The L_p-norm w.r.t. a distribution μ is a common choice.
- Later we see that μ might be the distribution induced by a behaviour policy.

Bellman Residual Minimization (Population Version)

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - T^\pi V\|_{p,\mu} = \|\mathrm{BR}(V)\|_{p,\mu} . \tag{7}$$

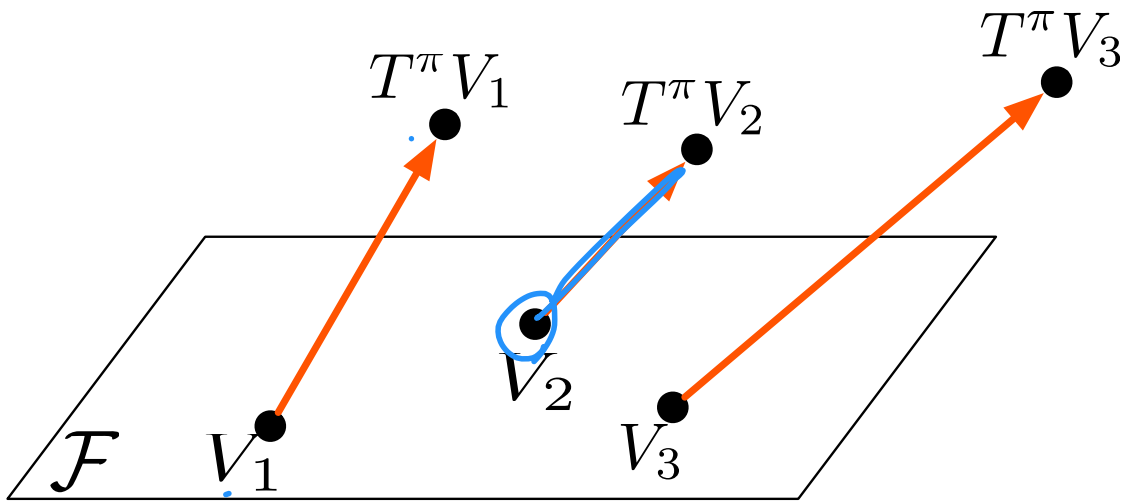
The value of p is often selected to be 2. This procedure is called the Bellman Residual Minimization (BRM).

The same procedure works for the action-value function Q with obvious changes.

Remark

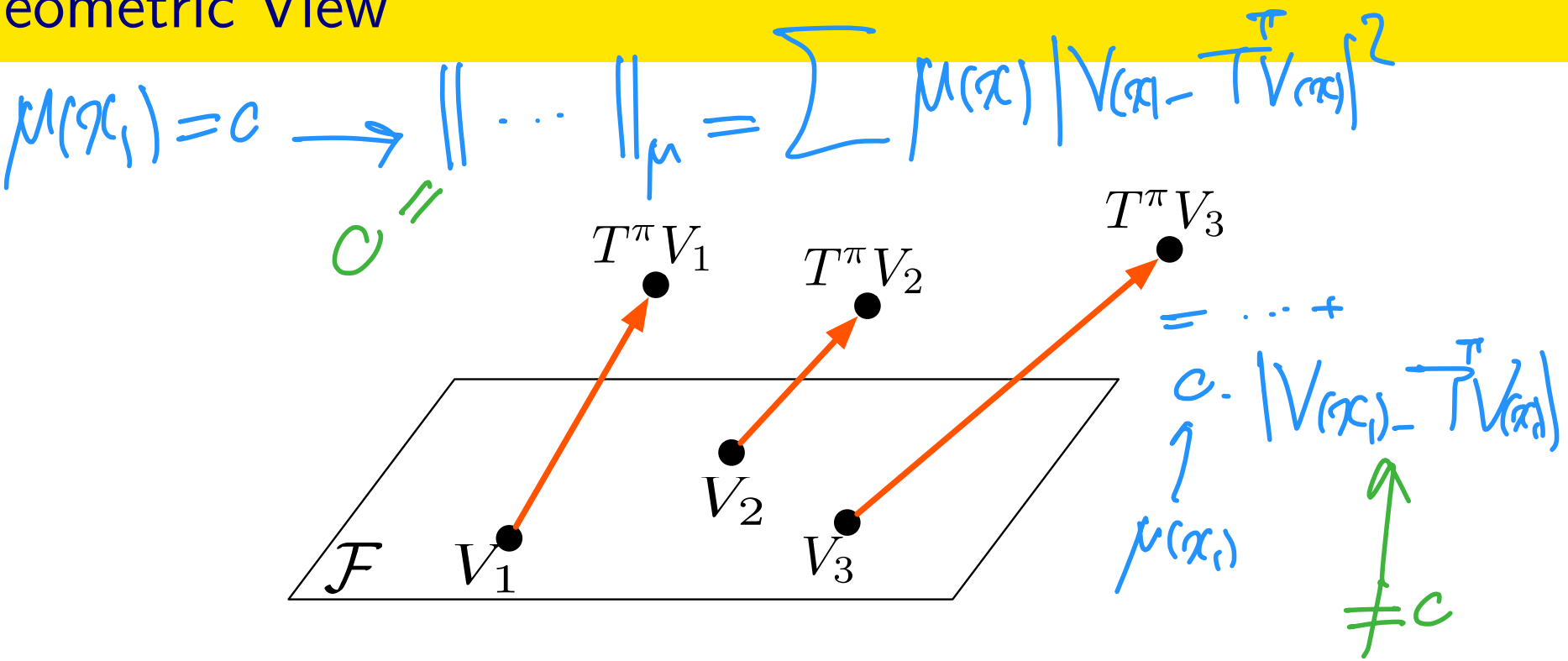
This procedure is different from AVI in that we do not mimic the iterative process of VI (which is convergent in the exact case without any FA), but instead directly go for the solution of the fixed-point equation.

Bellman Residual Minimization (Population Version): A Geometric View



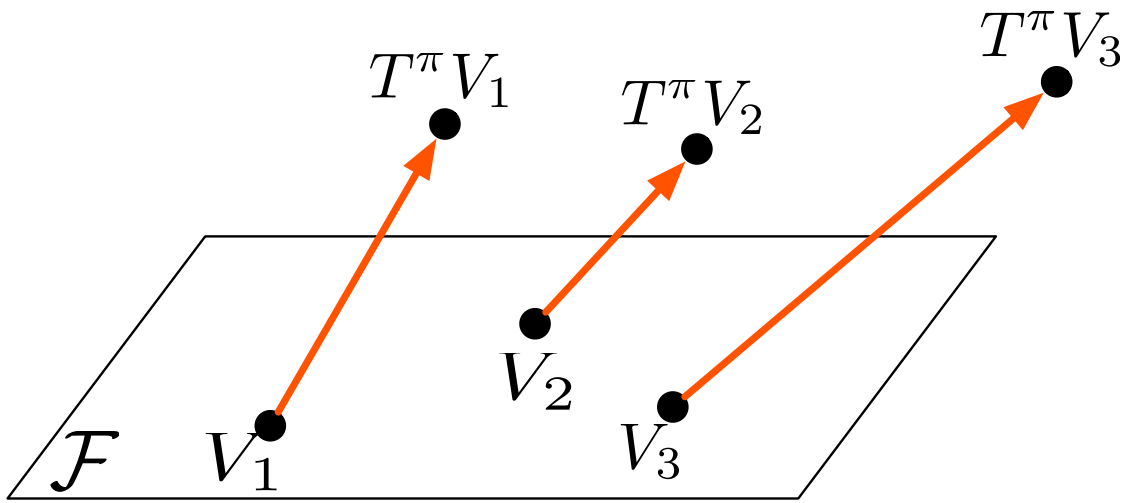
When $\mathcal{F}$ is the set of linear functions (1), its geometry is the subspace spanned by ϕ.
Given $V \in \mathcal{F}$, we apply T^π to it in order to get $T^\pi V$. In general, $T^\pi V$ is not within $\mathcal{F}$, so we visualize it with a point outside the plane.
BRM minimizes the distance $\|V - T^\pi V\|_{2,\mu}$ among all functions in $V \in \mathcal{F}$.

Bellman Residual Minimization (Population Version): A Geometric View



If there exists a $V \in \mathcal{F}$ that makes $\|V - T^\pi V\|_{2,\mu} = 0$, and if we assume that $\mu(x) > 0$ for all $x \in \mathcal{X}$, we can conclude that $V(x) = (T^\pi V)(x)$ for $x \in \mathcal{X}$ (a.s.). (Q: Why $\mu(x) > 0$ is needed?)
This is the Bellman equation, so its solution is $V = V^\pi$.

Bellman Residual Minimization (Population Version): A Geometric View



In general, the error is not zero, so the minimizer V of (7) is not the value function V^π.
Nevertheless, it still has some good approximation properties.

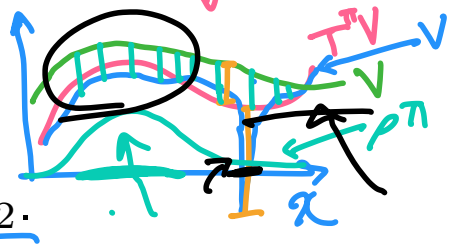
Bellman Error-based Error Bound: Supremum Norm

- We can relate the error in approximating a value function to the Bellman error:

$$\|V - V^\pi\|_\infty \leq \frac{\|V - T^\pi V\|_\infty}{1-\gamma}.$$

  Recall that we proved this in the Structural Properties of MDP lecture.

- The Bellman error is a surrogate loss.
  - Instead of minimizing $\|V - V^\pi\|$, we minimize its upper bound $\|V - T^\pi V\|$.
  - If the upper bound is small, the quantity of our interest is small too.
- Caveat: This is the supremum norm.
  - Very conservative and unforgiving
  - In ML, we often minimize L_p-norms, e.g., L_2.



- We can provide a similar bound using a stationary distribution of a policy π.

Stationary Distribution of Policy π



Definition (Intuitive)

The stationary (or invariant) distribution of a policy π is the distribution that does not change as we follow π.

- We initiate the agent at $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$.
- The agent follows π and gets to $X_2 \sim \mathcal{P}^\pi(\cdot|X_1)$.
- The probability distribution of X_2 being in a (measurable) set B is

$$\mathbb{P}\{X_2 \in B\} = \int \rho(\mathrm{d}x)\mathcal{P}^\pi(B|x),$$

or for countable state space, the probability of being in state y is

$$\mathbb{P}\{X_2 = y\} = \sum_{x \in \mathcal{X}} \rho(x)\mathcal{P}^\pi(y|x).$$

Stationary Distribution of Policy π

- If the distribution of X_1 and X_2 are both ρ^π, we say that ρ^π is the stationary distribution induced by π.
- It would be the distribution of $X_3, X_4, \ldots$ too.
- If X_1 and X_2 are both at the stationary distribution, we have $\mathbb{P}\{X_1 = y\} = \mathbb{P}\{X_2 = y\}$ for any $y \in \mathcal{X}$,

$$\mathbb{P}\{X_1 = y\} = \rho^\pi(y) = \sum_{x \in \mathcal{X}} \mathcal{P}^\pi(y|x)\rho^\pi(x) = \mathbb{P}\{X_2 = y\}, \tag{8}$$

or

$$\rho^\pi(B) = \int \rho^\pi(\mathrm{d}x)\mathcal{P}^\pi(B|x). \tag{9}$$

Stationary Distribution of Policy π

- For countable state spaces, we can write it in the matrix form.
- If we denote $\mathcal{P}^\pi$ by an $n \times n$ matrix with $[\mathcal{P}^\pi]_{xy} = \mathcal{P}^\pi(y|x)$, we have

$$\rho^\pi(y) = \sum_x \mathcal{P}^\pi_{xy} \rho^\pi_x, \qquad \forall y \in \mathcal{X}$$

so

$$\rho^{\pi\top} = \rho^{\pi\top} \mathcal{P}^\pi. \tag{10}$$

- The distribution ρ^π is the left eigenvector corresponding to eigenvalue with value 1 of matrix $\mathcal{P}^\pi$ (or likewise, the right eigenvector of $\mathcal{P}^{\pi\top}$).

Stationary Distribution of Policy π

- Under certain conditions, a Markov chain induced by π converges to ρ^π, even if the initial distribution is not ρ^π.
- For any $\mu \in \mathcal{M}(\mathcal{X})$, we have that

$$\mu(\mathcal{P}^\pi)^k \to \rho^\pi.$$

Stationary Distribution of Policy π

Lemma

The Bellman operator T^π is a γ-contraction w.r.t. $\|\cdot\|_{2,\rho^\pi}$.

Bellman Error-based Error Bound w.r.t. the Stationary Distribution

Proposition

Let ρ^π be the stationary distribution of $\mathcal{P}^\pi$. For any $V \in \mathcal{B}(\mathcal{X})$ and $p \geq 1$, we have

$$\|V - V^\pi\|_{1,\rho^\pi} \leq \frac{\|V - T^\pi V\|_{p,\rho^\pi}}{1-\gamma}.$$

Remark

This is similar to $\|V - V^\pi\|_\infty \leq \frac{\|V - T^\pi V\|_\infty}{1-\gamma}$. but is w.r.t. the stationary distribution ρ^π.

Bellman Error-based Error Bound w.r.t. the Stationary Distribution (Proof)

For any V, we have that

$$\begin{aligned} V - V^\pi &= V - T^\pi V + T^\pi V - V^\pi \\ &= (V - T^\pi V) + (T^\pi V - T^\pi V^\pi). \end{aligned} \tag{11}$$

The second term on the RHS, evaluated at a state x, is

$$(T^\pi V)(x) - (T^\pi V^\pi)(x) = \gamma \int \mathcal{P}^\pi(\mathrm{d}y|x)(V(y) - V^\pi(y)).$$

Bellman Error-based Error Bound w.r.t. the Stationary Distribution (Proof)

Take the absolute value, use the obtained form of the second term, and integrate w.r.t. ρ^π:

$$\int |V(x) - V^\pi(x)| \, \mathrm{d}\rho^\pi(x) \leq \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \gamma \int \mathrm{d}\rho^\pi(x) \left| \int \mathcal{P}^\pi(\mathrm{d}y|x)(V(y) - V^\pi(y)) \right| .$$

By Jensen's inequality, we have

$$\int |V(x) - V^\pi(x)| \, \mathrm{d}\rho^\pi(x) \leq \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \gamma \int \mathrm{d}\rho^\pi(x) \mathcal{P}^\pi(\mathrm{d}y|x) \, |V(y) - V^\pi(y)| .$$

Bellman Error-based Error Bound w.r.t. the Stationary Distribution (Proof)

Because ρ^π is the stationary distribution, the second integral in the RHS can be simplified as

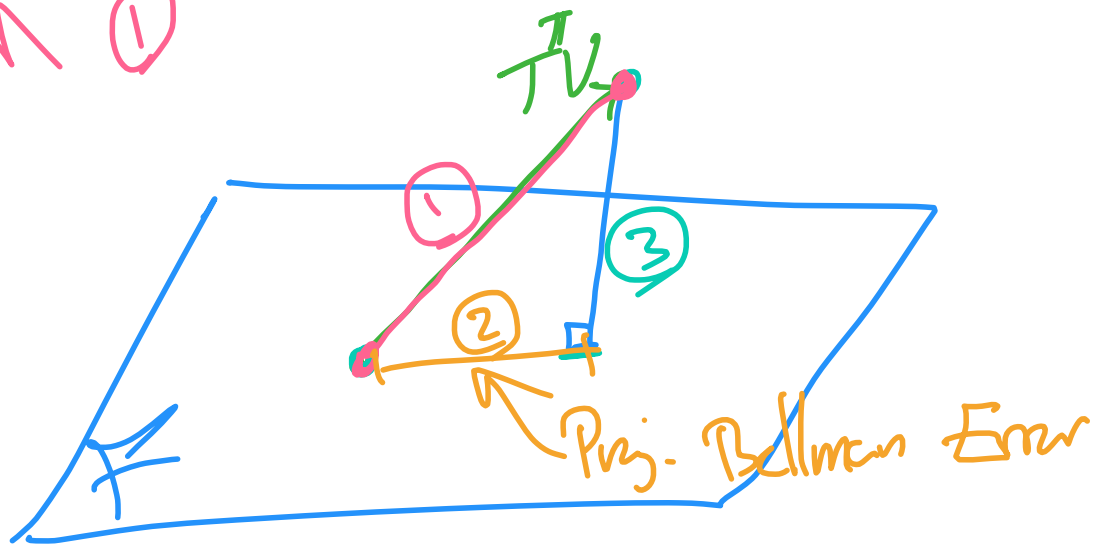
$$\int \mathrm{d}\rho^\pi(x)\mathcal{P}^\pi(\mathrm{d}y|x)\,|V(y) - V^\pi(y)| = \int \mathrm{d}\rho^\pi(y)\,|V(y) - V^\pi(y)|\,.$$

So

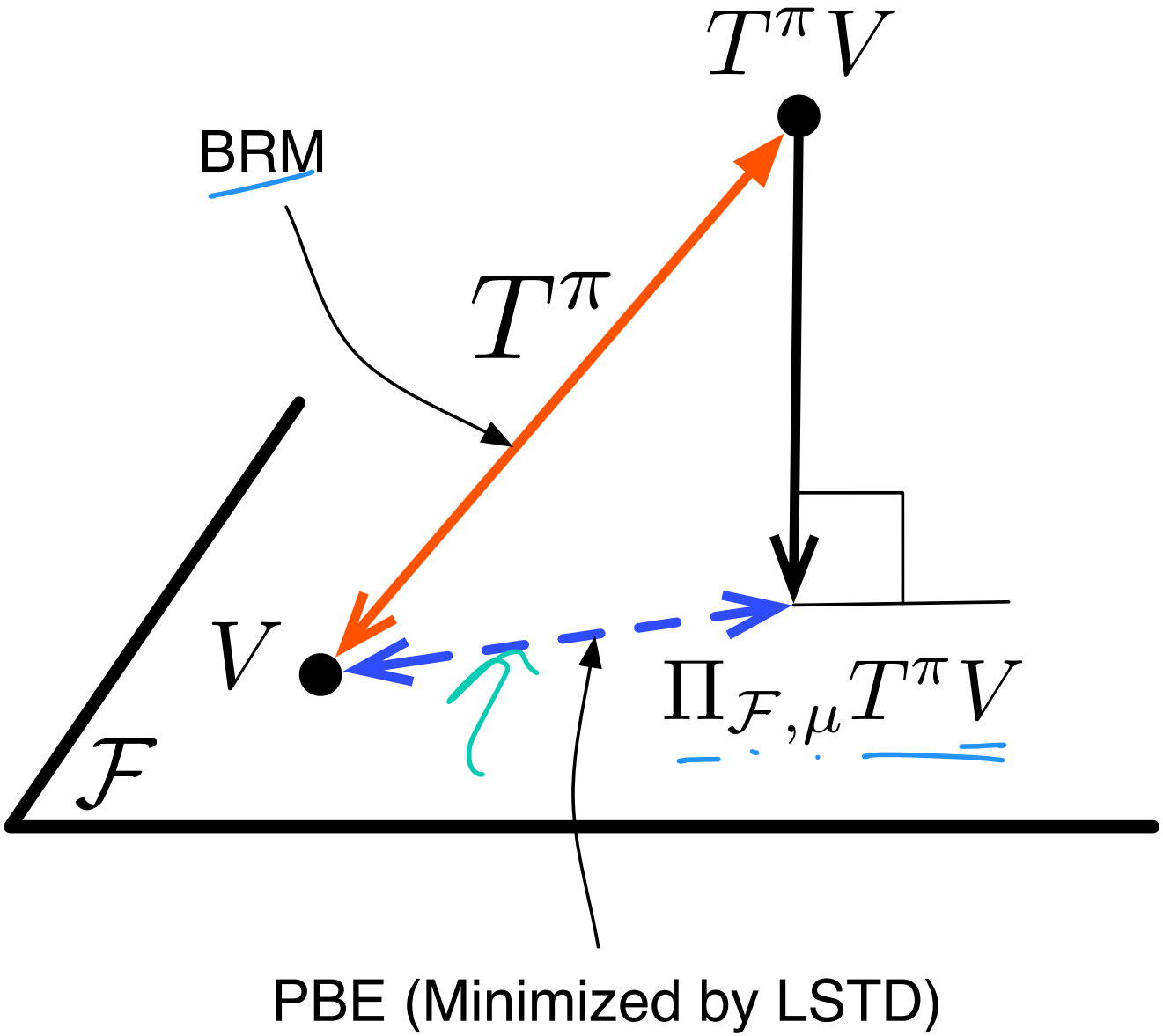
$$\|V - V^\pi\|_{1,\rho^\pi} \leq \|V - T^\pi V\|_{1,\rho^\pi} + \gamma\,\|V - V^\pi\|_{1,\rho^\pi}\,.$$

After re-arranging, we get the result for $p = 1$.
By Jensen's inequality, we have that $\|V - T^\pi V\|_{1,\rho^\pi} \leq \|V - T^\pi V\|_{p,\rho^\pi}$, for any $p \geq 1$.

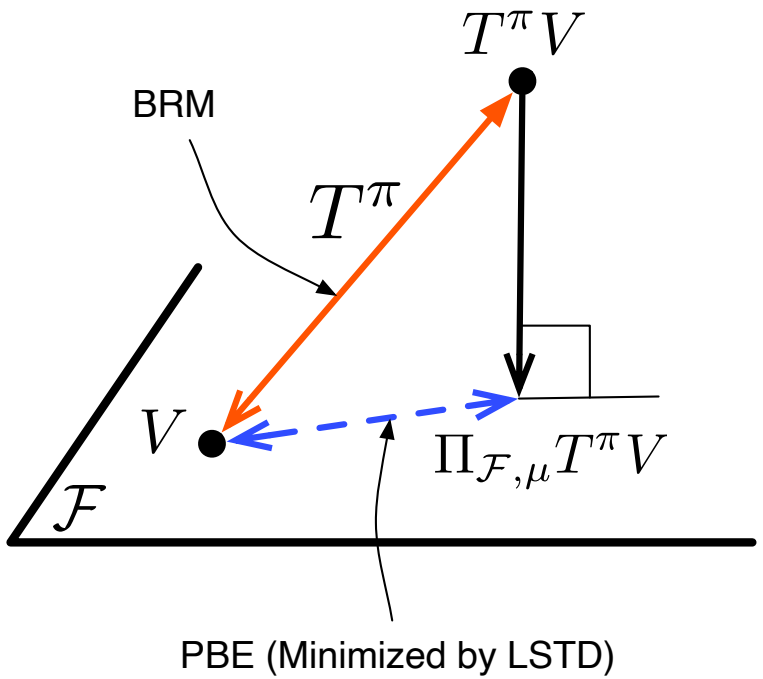
Projected Bellman Error



Projected Bellman Error (Population Version)

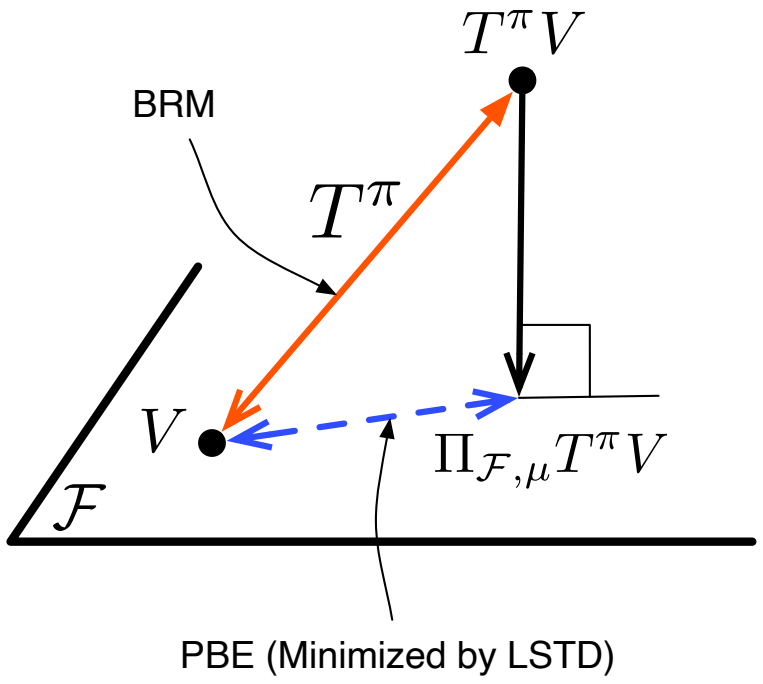


Projected Bellman Error (Population Version)



Main idea: The distance between a value function $V \in \mathcal{F}$ and the projection of $T^\pi V$ onto $\mathcal{F}$ should be made small.

Projected Bellman Error (Population Version)



We find a $V \in \mathcal{F}$ such that

$$V = \Pi_{\mathcal{F},\mu} T^\pi V, \tag{12}$$

where $\Pi_{\mathcal{F},\mu}$ is the projection operator onto $\mathcal{F}$.

Projection Operator

The projection operator $\Pi_{\mathcal{F},\mu}$ is a linear operator that takes $V \in \mathcal{B}(\mathcal{X})$ and maps it to the closest point on $\mathcal{F}$, measured according to its $L_2(\mu)$ norm.

$$\Pi_{\mathcal{F},\mu} V \triangleq \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - V \right\|_{2,\mu}.$$

If the choice of distribution μ is clear from the context, we may omit it.
Some properties:



- $\Pi_{\mathcal{F},\mu} V \in \mathcal{F}$.
- If $V \in \mathcal{F}$, we have $\Pi_{\mathcal{F},\mu} V = V$.
- The projection operator onto a subspace (also a closed convex set) is a non-expansion, i.e., $\left\| \Pi_{\mathcal{F},\mu} V_1 - \Pi_{\mathcal{F},\mu} V_2 \right\|_{2,\mu} \leq \left\| V_1 - V_2 \right\|_{2,\mu}$.

Projected Bellman Error (Population)

We can define a loss function based on $V = \Pi_{\mathcal{F}} T^{\pi} V$ (12).
We can use different norms. A common choice is the $L_2(\mu)$-norm:

$$\|V - \Pi_{\mathcal{F}} T^{\pi} V\|_{2,\mu} . \tag{13}$$

This is called Projected Bellman Error or its squared valued as Mean Squared Projected Bellman Error (MSPBE).
We find the value function by solving the following optimization problem:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - \Pi_{\mathcal{F}} T^{\pi} V\|_{2,\mu} . \tag{14}$$

Projected Bellman Error (Population)

As $V \in \mathcal{F}$,

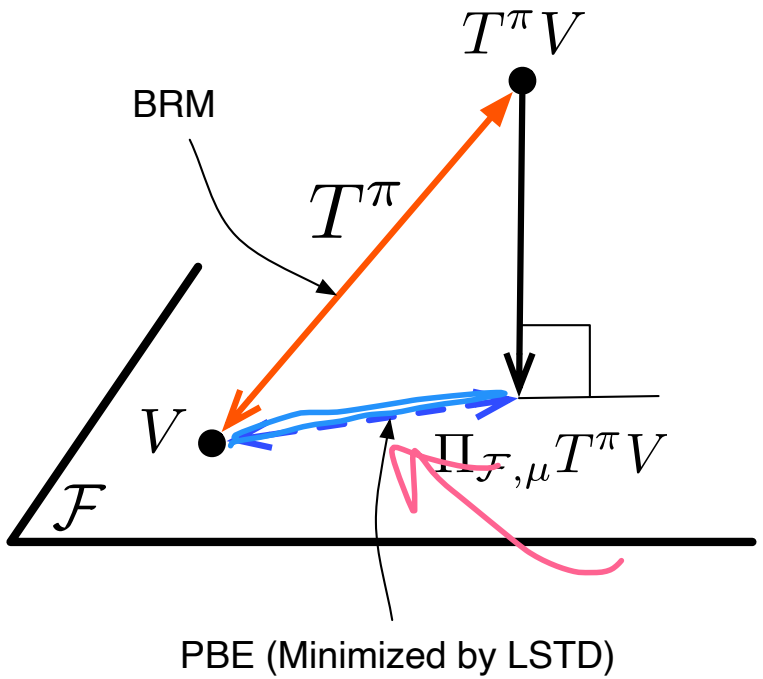
$$\begin{aligned} V - \Pi_{\mathcal{F},\mu} T^\pi V &= \Pi_{\mathcal{F},\mu} V - \Pi_{\mathcal{F},\mu} T^\pi V \\ &= \Pi_{\mathcal{F},\mu}(V - T^\pi V) \\ &= -\Pi_{\mathcal{F},\mu}(\mathsf{BR}(V)). \end{aligned}$$

So the loss is

$$\|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu} = \|\Pi_{\mathcal{F},\mu}(\mathsf{BR}(V))\|_{2,\mu}$$

The norm of the projection of the Bellman residual onto $\mathcal{F}$.

Bellman Residual Minimization vs Projected Bellman Error



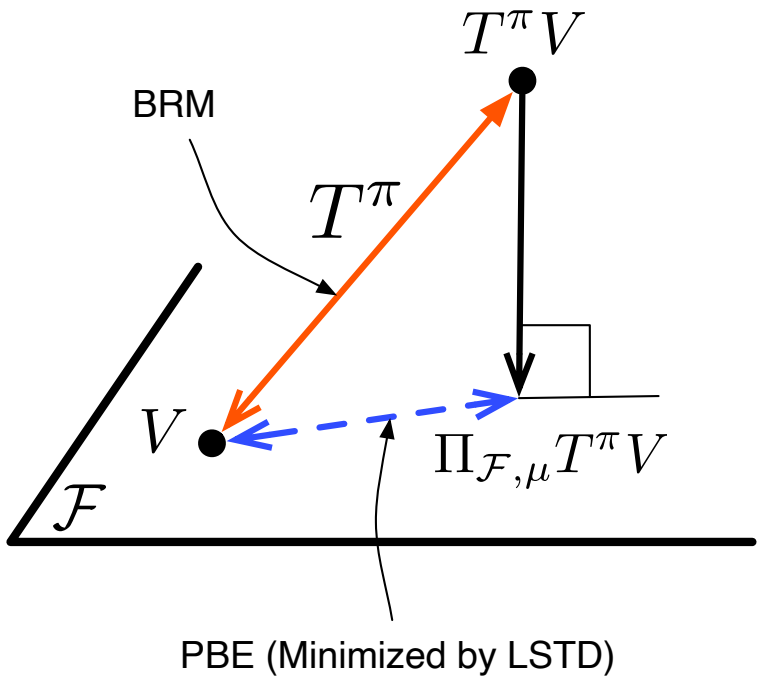
Bellman Residual Minimization:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - T^\pi V\|_{p,\mu} = \|\mathsf{BR}(V)\|_{p,\mu} .$$

Projected Bellman Error:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu} = \|\Pi_{\mathcal{F},\mu}(\mathsf{BR}(V))\|_{2,\mu}$$

Coupled (Nested) Formulation of Projected Bellman Error



We can think of the PBE as simultaneously solving these two coupled (or nested) optimization problems:

$$V \leftarrow \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - \tilde{V}(V') \right\|_{2,\mu}^2 ,$$
$$\tilde{V}(V') \leftarrow \operatorname*{argmin}_{V'' \in \mathcal{F}} \left\| V'' - T^\pi V' \right\|_{2,\mu}^2 . \tag{15}$$

Coupled (Nested) Formulation of Projected Bellman Error

- If $\mathcal{F}$ is a linear function space, the projection has a closed-form solution.
- For more general spaces, the solution may not be simple.
- Regularized variants: suitable for avoiding overfitting when $\mathcal{F}$ is a very large function space.

$$V \leftarrow \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - \tilde{V}(V') \right\|_{2,\mu}^2 + \lambda J(V'),$$

$$\tilde{V}(V') \leftarrow \operatorname*{argmin}_{V'' \in \mathcal{F}} \left\| V'' - T^\pi V' \right\|_{2,\mu}^2 + \lambda J(V'').$$

Solving PBE: Several Surprisingly Disparate Approaches

There are different approaches to solve (14), some of which may not appear to be related at first glance.
Let us look at the abstract problem of solving a linear system of equations before getting back to this.

Solving $Ax \approx b$

Suppose that we want to solve a linear system of equations

$$Ax \approx b, \tag{16}$$

with $A \in \mathbb{R}^{N \times d}$, $x \in \mathbb{R}^d$, and $b \in \mathbb{R}^N$ ($N \geq d$).
When $N > d$, this is an overdetermined system so the equality may not be satisfied.

Solving $Ax \approx b$: Optimization Approach

Formulate it as an optimization problem:

$$x^* \leftarrow \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|Ax - b\|_2^2 = \frac{1}{2}(Ax - b)^\top (Ax - b). \tag{17}$$

We can use our favourite numerical optimizer to solve it, e.g., Gradient Descent (GD).
As the gradient of $\frac{1}{2}(Ax - b)^\top (Ax - b)$ is

$$A_{d \times N}^\top (A_{N \times d} x - b),$$

the GD procedure would be

$$x_{k+1} \leftarrow x_k - \alpha A^\top (Ax_k - b).$$

We can use more advanced optimization techniques too. This approach finds an x^* that minimizes the squared error loss function.

Solving $Ax \approx b$: Direct Approach

Solve for the zero of the gradient:

$$A^\top Ax = A^\top b \Rightarrow x^* = (A^\top A)^{-1} A^\top b, \tag{18}$$

assuming the invertibility of $A^\top A$.
For this approach, we need to have a method to invert the matrix $A^\top A$.

Solving $Ax \approx b$: Fixed-Point Iteration

We can rewrite $Ax = b$ as

$$(\mathbf{I} - A)x + b = x.$$

Suppose $N = d$. This is of the form of a fixed-point equation

$$Lx = x$$

with $L : \mathbb{R}^d \to \mathbb{R}^d$ being the mapping

$$L : x \mapsto (\mathbf{I} - A)x + b.$$

Solving $Ax \approx b$: Fixed-Point Iteration

If L is a contraction mapping (not always the case), by the Banach fixed point theorem, the iterative procedure

$$x_{k+1} \leftarrow Lx_k = (\mathbf{I} - A)x_k + b \tag{19}$$

converges to x^*, the solution of $Ax^* = b$.
It is also possible to define a slightly modified procedure of

$$x_{k+1} \leftarrow (1-\alpha)x_k + \alpha Lx_k. \tag{20}$$

This is similar to the iterative procedure we saw before in the SA (without noise).

Least Squares Temporal Difference Learning (Population)

Instead of minimizing $\|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu}$ over value functions $V \in \mathcal{F}$, we provide a direct solution similar to (18).

$\mathcal{F}$: a linear FA with basis functions (or features) $\phi_1, \ldots, \phi_p$.

$$\mathcal{F} = \left\{ x \mapsto \phi(x)^\top w \ : \ w \in \mathbb{R}^p \right\}.$$

Goal: Find a value function that satisfies

$$V(x) = \left(\Pi_{\mathcal{F},\mu} T^\pi V\right)(x), \qquad \forall x \in \mathcal{X}, \tag{21}$$

where V is restricted to be in $\mathcal{F}$.

Least Squares Temporal Difference Learning (Population)

We assume that $\mathcal{X}$ is finite and has N states, potentially much larger than p.

Each ϕ_i $(i = 1, \ldots, p)$ is an N-dimensional vector, in the form of

$$\phi_i = \begin{bmatrix} \phi_i(x_1) \\ \vdots \\ \phi_i(x_N) \end{bmatrix}$$

The matrix of concatenating all features, $\Phi \in \mathbb{R}^{N \times p}$, is:

$$\Phi = \begin{bmatrix} \phi_1 & \cdots & \phi_p \end{bmatrix}.$$

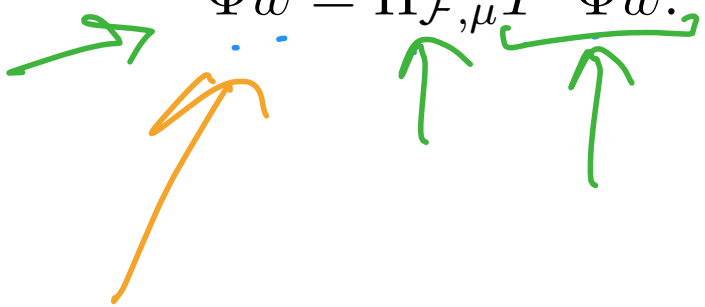
The value function corresponding to a weight $w \in \mathbb{R}^p$ is

$$V_{N \times 1} = \Phi_{N \times p} w_p.$$

Least Squares Temporal Difference Learning (Population)

Solving $V = (\Pi_{\mathcal{F},\mu} T^\pi V)$ when $V = V(w) = \Phi w \in \mathcal{F}$ means that we have to find a $w \in \mathbb{R}^p$ such that

$$\Phi w = \Pi_{\mathcal{F},\mu} T^\pi \Phi w. \tag{22}$$



Least Squares Temporal Difference Learning (Population)

The μ-weighted inner product between $V_1, V_2 \in \mathbb{R}^N$:

$$\langle V_1 , V_2 \rangle_\mu = \sum_x V_1(x) V_2(x) \mu(x) = V_1^\top M V_2, \tag{23}$$

with $M = \mathrm{diag}(\mu)$.

The $L_2(\mu)$-norm:

$$\|V\|_{2,\mu}^2 = \langle V , V \rangle_\mu = \sum_{x \in \mathcal{X}} |V(x)|^2 \mu(x) = V^\top M V.$$

Least Squares Temporal Difference Learning (Population)

The projection operator onto a linear $\mathcal{F}$ would be

$$\Pi_{\mathcal{F},\mu} V = \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - V \right\|_{2,\mu}^2 .$$

We can identify the projected function $\hat{V} = \Phi w^+ = \Pi_{\mathcal{F},\mu} V$ with the weight w^+ minimizing the following:

$$\operatorname*{argmin}_{w \in \mathbb{R}^p} \left\| \Phi w - V \right\|_{2,\mu}^2 = \operatorname*{argmin}_{w \in \mathbb{R}^p} (\Phi w - V)^\top M (\Phi w - V).$$

Taking the derivative and setting it to zero:

$$\Phi^\top M (\Phi w - V) = 0 \Rightarrow w^+ = (\Phi^\top M \Phi)^{-1} \Phi^\top M V$$

(assuming that $\Phi^\top M \Phi$ is invertible.)
The projected function is Φw^+, i.e.,

$$\Pi_{\mathcal{F},\mu} V = \Phi (\Phi^\top M \Phi)^{-1} \Phi^\top M V. \tag{24}$$

Least Squares Temporal Difference Learning (Population)

We have

$$(T^\pi \Phi w)_{N\times 1} = r^\pi_{N\times 1} + \gamma \mathcal{P}^\pi_{N\times N} \Phi_{N\times p} w_p.$$

Combining all these:

$$\Phi w = \left[\Phi(\Phi^\top M \Phi)^{-1}\Phi^\top M\right]\left[r^\pi + \gamma \mathcal{P}^\pi \Phi w\right]. \tag{25}$$

Two approaches to solve this:

- Solve directly, cf (18).
- Fixed-point iteration, cf (20).

Least Squares Temporal Difference Learning (Population)

Multiply both sides of (25) by $\Phi^\top M$ and simplify:

$$\begin{aligned}\Phi^\top M\Phi w &= \Phi^\top M\Phi(\Phi^\top M\Phi)^{-1}\Phi^\top M\left[r^\pi + \gamma\mathcal{P}^\pi\Phi w\right] \\ &= \Phi^\top M\left[r^\pi + \gamma\mathcal{P}^\pi\Phi w\right]. \\ \Rightarrow \Phi^\top M\left[r^\pi + \gamma\mathcal{P}^\pi\Phi w - \Phi w\right] &= 0. \qquad (26)\end{aligned}$$

Re-arrange to

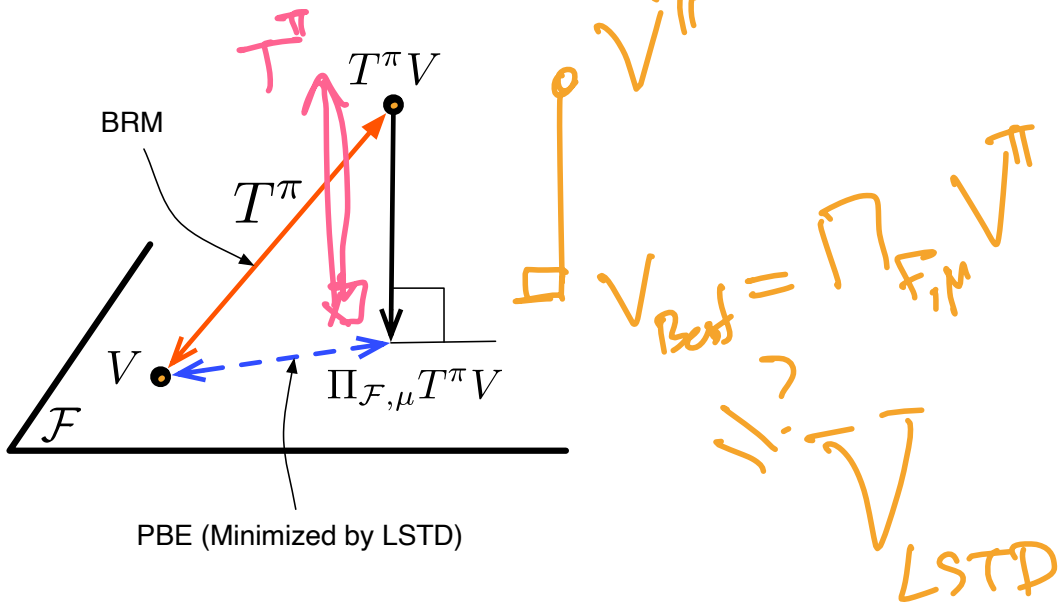
$$\left[\Phi^\top M\Phi - \gamma\Phi^\top M\mathcal{P}^\pi\Phi\right] w = \Phi^\top M r^\pi.$$

Solving for w:

$$w = \left[\Phi^\top M(\Phi - \gamma\mathcal{P}^\pi\Phi)\right]^{-1}\Phi^\top M r^\pi. \qquad (27)$$

This is the population version of the Least Squares Temporal Difference (LSTD) method.

Least Squares Temporal Difference Learning (Population) – Geometric Intuition



- $\Phi^\top M \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w \right] = 0$
- $\langle V_1 , V_2 \rangle_\mu = V_1^\top M V_2$

$$\langle \phi_i , T^\pi V(w) - V(w) \rangle_\mu = \langle \phi_i , \mathsf{BR}(V(w)) \rangle_\mu = 0, \qquad \forall i = 1, \ldots, p.$$

LSTD finds a w such that the Bellman Residual is orthogonal to the basis of $\mathcal{F}$.

Fixed Point Iteration for Projected Bellman Operator – Approach #1

- We design two iterative approaches for finding the fixed-point of $\Pi_{\mathcal{F},\mu} T^\pi$, see (12).
- We attempt to design methods that look like an SA iteration, so when we deal with the samples, instead of the true model, they can handle the noise.
- We specifically focus on the case when the distribution μ is the stationary distribution ρ^π of π.

Fixed Point Iteration for Projected Bellman Operator – Approach #1

Consider

$$\hat{V}_{k+1} \leftarrow (1-\alpha)\hat{V}_k + \alpha \Pi_{\mathcal{F},\rho^\pi} T^\pi \hat{V}_k, \tag{28}$$

with $0 < \alpha \leq 1$.
A fixed-point iterative method with the operator

$$L : V \mapsto \left[(1-\alpha)\mathbf{I} + \alpha \Pi_{\mathcal{F},\rho^\pi} T^\pi\right] V.$$

This operator is a contraction w.r.t. $L_2(\rho^\pi)$:

$$\begin{aligned} \left\|LV_1 - LV_2\right\|_{2,\rho^\pi} \leq & (1-\alpha)\left\|V_1 - V_2\right\|_{2,\rho^\pi} \\ & + \alpha \left\|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_1 - \Pi_{\mathcal{F},\rho^\pi} T^\pi V_2\right\|_{2,\rho^\pi}. \end{aligned} \tag{29}$$

Fixed Point Iteration for Projected Bellman Operator – Approach #1

- The projection operator $\Pi_{\mathcal{F},\rho^\pi}$ is non-expansive w.r.t. the $L_2(\rho^\pi)$
- The Bellman operator T^π is γ-contraction w.r.t. the same norm (Lemma 2).

$$\begin{aligned}\left\|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_1 - \Pi_{\mathcal{F},\rho^\pi} T^\pi V_2\right\|_{2,\rho^\pi} &\leq \left\|T^\pi V_1 - T^\pi V_2\right\|_{2,\rho^\pi} \\ &\leq \gamma \left\|V_1 - V_2\right\|_{2,\rho^\pi} .\end{aligned}$$

This, together with (29), shows that

$$\left\|LV_1 - LV_2\right\|_{2,\rho^\pi} \leq [(1-\alpha) + \alpha\gamma] \left\|V_1 - V_2\right\|_{2,\rho^\pi} .$$

If $0 < \alpha \leq 1$, we also have $(1-\alpha) + \alpha\gamma < 1$, therefore, L is a contraction operator.

Fixed Point Iteration for Projected Bellman Operator – Approach #1

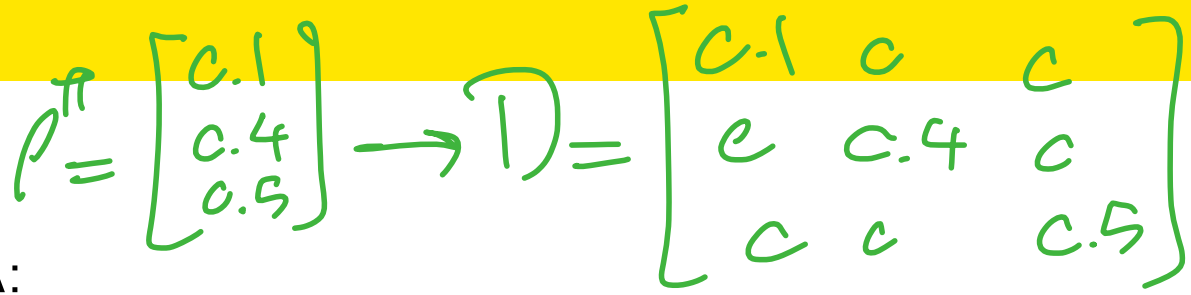


- The iterative method (28) is going to be convergent.

$$\hat{V}_{k+1} \leftarrow (1-\alpha)\hat{V}_k + \alpha \Pi_{\mathcal{F},\rho^\pi} T^\pi \hat{V}_k$$

- Note: Its projection operator is w.r.t. $\|\cdot\|_{2,\rho^\pi}$. The convergence property may not hold for other $\mu \neq \rho^\pi$.

Fixed Point Iteration for Projected Bellman Operator – Approach #1



Let us use a linear FA:

$$\hat{V}_k = \Phi w_k.$$

We use the explicit formula (24) for the projection operator $\Pi_{\mathcal{F},\rho^\pi}$.
We use $D = \mathrm{diag}(\rho^\pi)$, instead of M, in order to emphasize the dependence on ρ^π.
The iteration (28) can be written as

$$\hat{V}_{k+1} = \Phi w_{k+1} \leftarrow (1-\alpha)\Phi w_k + \alpha\Phi(\Phi^\top D^\pi \Phi)^{-1}\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right].$$

Fixed Point Iteration for Projected Bellman Operator – Approach #1

Multiply both sides by $\Phi^\top D^\pi$:

$$(\Phi^\top D^\pi \Phi) w_{k+1} \leftarrow (1-\alpha)(\Phi^\top D^\pi \Phi) w_k + \alpha (\Phi^\top D^\pi \Phi)(\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k \right].$$

Assuming that $\Phi^\top D^\pi \Phi$ is invertible:

$$w_{k+1} \leftarrow (1-\alpha) w_k + \alpha (\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k \right]. \tag{30}$$

This is a convergent iteration and converges to the fixed point of $\Phi w = \Pi_{\mathcal{F},\rho^\pi} T^\pi \Phi w$ (21).
This is the same as the LSTD's solution (27) with the distribution ρ^π.

Fixed Point Iteration for Projected Bellman Operator – Approach #1

$$w_{k+1} \leftarrow (1-\alpha)w_k + \alpha(\Phi^\top D^\pi \Phi)^{-1}\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right].$$

- This requires a one-time inversion of a $p \times p$ matrix $(\Phi^\top D^\pi \Phi) = \sum_x \rho(x)\phi(x)\phi^\top(x)$, which is $O(p^3)$ operation.
- A matrix-vector multiplication at every time step $O(p^2)$.
- When we move to the online setting, where this matrix itself is updated as every new data point arrives, a naive approach of updating the matrix and re-computing its inverse, would be costly.

Fixed Point Iteration for Projected Bellman Operator – Approach #2

From (26):

$$\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w\right] = 0. \tag{31}$$

The same solution as the LSTD solution.

- If $Lw = 0$, we also have $\alpha L w = 0$.
- Adding an identity to both sides does not change the equation

$$w + \alpha L w = w.$$

- This is in the form of a fixed-point equation for a new operator

$$L' : w \mapsto (\mathbf{I} + \alpha L) w.$$

- The fixed point of L' is the same as the solution of $Lw = 0$.
- We may apply $w_{k+1} \leftarrow L' w_k = (\mathbf{I} + \alpha L) w_k$, assuming L' is a contraction.

Fixed Point Iteration for Projected Bellman Operator – Approach #2

If we choose $L : w \mapsto \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w\right]$, we get the following iterative procedure, which is somewhat similar to (20):

$$\begin{aligned} w_{k+1} \leftarrow w_k + \alpha \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k\right] \\ = (\mathbf{I} - \alpha A) w_k + \alpha \Phi^\top D^\pi r^\pi, \end{aligned} \tag{32}$$

with $A = \Phi^\top D^\pi (\mathbf{I} - \gamma \mathcal{P}^\pi) \Phi$.

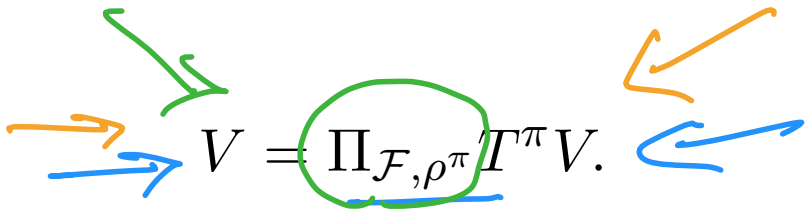
Remark

This iterative procedure is not a convex combination of $\Pi_{\mathcal{F},\rho^\pi} T^\pi$ with the identity matrix, as (28) was, so the condition for convergence does not follow from what we had before.
Despite that, we can show that for small enough α, it is convergent.

Error Bound on the LSTD Solution



Suppose that we find



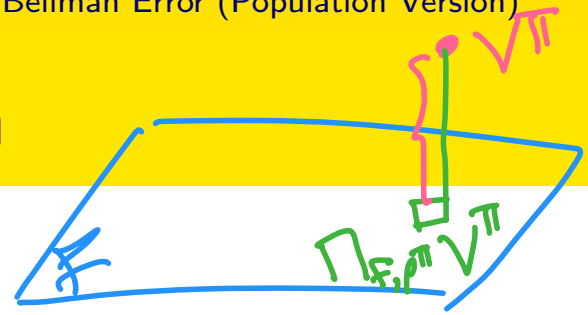
$$V = \Pi_{\mathcal{F},\rho^\pi} T^\pi V.$$

For the linear FA, the LSTD method (population) (27) and the fixed point iterations (28) and (32) find this solution. Let us call this the TD solution V_{TD}.

Q: How close is this value function to the true value function V^π?



Error Bound on the LSTD Solution



- If the value function space $\mathcal{F}$ cannot represent V^π precisely, which is often the case under function approximation, we cannot expect to have a small error.
- The smallest error we can hope is $\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|$.
- The TD solution is not as close to V^π as the projection of V^π onto $\mathcal{F}$, but it can be close to that.

Proposition

If ρ^π is the stationary distribution of π, we have

$$\|V_{TD} - V^\pi\|_{2,\rho^\pi} \leq \frac{\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|_{2,\rho^\pi}}{\sqrt{1-\gamma^2}}.$$

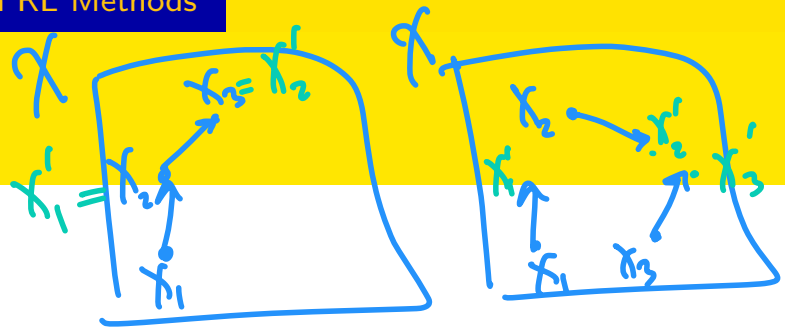
$\gamma = 0.9 \rightarrow \sqrt{1 - 0.81}$

Batch RL Methods

From Population-based Methods to Sample-based Methods

- We use ideas developed in the previous section to develop RL algorithms that work with function approximators.
- Key step: Finding an empirical version of the relevant quantities and estimate them using data.
- For example, many of the aforementioned methods require the computation of TV. If the model is not known, this cannot be computed. We have to come up with a procedure that estimates TV based only on data.

Batch RL



We consider the batch (or offline) data setting.

- The data is already collected, and we are interested in using it to estimate quantities such as Q^π or Q^*.
- Suppose that we have

$$\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n, \tag{33}$$

with $(X_i, A_i) \sim \mu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$, and $X_i' \sim \mathcal{P}(\cdot|X_i, A_i)$ and $R_i \sim \mathcal{R}(\cdot|X_i, A_i)$.

- The data could be generated by following a behaviour policy π_b and having trajectories in the form of $(X_1, A_1, R_1, X_2, A_2, R_2, \dots)$. In this case, $X_t' = X_{t+1}$.

Batch RL

- In the batch setting, the agent does not interact with the environment while it is computing Q^π, Q^*, etc.
- This can be contrasted with the online method such as TD or Q-Learning, where the agent updates its estimate of the value function as each data point arrives.
- The boundary between the batch and online methods is blurry.
- A method may collect a batch of data, process them, and then collect a new batch of data, possibly based on a policy resulting from the previous batch processing computation.

Value Function Approximation Given the Monte Carlo Estimates

A batch of data:

$$\mathcal{D}_n = \{(X_i, A_i, G^\pi(X_i, A_i))\}_{i=1}^n,$$

with $G^\pi(X_i, A_i)$ being a return of being at state X_i, taking action A_i, and following the policy π afterwards.
The distribution of $(X_i, A_i) \sim \mu$.
The return can be obtained using the initial-state only MC by selecting $(X_i, A_i) \sim \mu$ and then following π until the end of episode (in the episodic case).
Q: Any other approach?

Value Function Approximation Given the Monte Carlo Estimates

Population loss function:

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - Q^\pi\|_{2,\mu}^2 . \tag{34}$$

Two differences with the current setup:

- We do not have a direct access to the distribution μ and only have samples from it.
- We do not know Q^π itself and we only have unbiased estimates G^π at a finite number of data points.

Value Function Approximation Given the Monte Carlo Estimates

Having access to unbiased noisy estimate of Q^π does not change the solution of the minimization problem.
For any Q, we can decompose:

$$\mathbb{E}\left[|Q(X,A) - G^\pi(X,A)|^2\right] =$$
$$\mathbb{E}\left[|Q(X,A) - Q^\pi(X,A) + Q^\pi(X,A) - G^\pi(X,A)|^2\right] =$$
$$\mathbb{E}\left[|Q(X,A) - Q^\pi(X,A)|^2\right] +$$
$$\mathbb{E}\left[|Q^\pi(X,A) - G^\pi(X,A)|^2\right] +$$
$$2\mathbb{E}\left[(Q(X,A) - Q^\pi(X,A))\,(Q^\pi(X,A) - G^\pi(X,A))\right].$$

The first term is $\|Q - Q^\pi\|_{2,\mu}^2$.
The second term is $\mathbb{E}\left[\mathrm{Var}\left[G^\pi(X,A) \mid X, A\right]\right]$.

Value Function Approximation Given the Monte Carlo Estimates

The inner product term:

$$\mathbb{E}\left[\left(Q(X,A) - Q^{\pi}(X,A)\right)\left(Q^{\pi}(X,A) - G^{\pi}(X,A)\right)\right] =$$
$$\mathbb{E}\left[\mathbb{E}\left[\left(Q(X,A) - Q^{\pi}(X,A)\right)\left(Q^{\pi}(X,A) - G^{\pi}(X,A)\right) \mid X, A\right]\right] =$$
$$\mathbb{E}\left[\left(Q(X,A) - Q^{\pi}(X,A)\right)\left(Q^{\pi}(X,A) - \mathbb{E}\left[G^{\pi}(X,A) \mid X, A\right]\right)\right] = 0.$$

As $\mathbb{E}\left[G^{\pi}(X,A) \mid X, A\right]$ is equal to $Q^{\pi}(X,A)$, the inside of second parenthesis in the last equality is zero. So the value of this term is zero.

Value Function Approximation Given the Monte Carlo Estimates

$$\underset{Q\in\mathcal{F}}{\operatorname{argmin}}\,\mathbb{E}\left[|Q(X,A)-G^{\pi}(X,A)|^2\right]=$$

$$\underset{Q\in\mathcal{F}}{\operatorname{argmin}}\left\{\mathbb{E}\left[|Q(X,A)-Q^{\pi}(X,A)|^2\right]+\mathbb{E}\left[\operatorname{Var}\left[G^{\pi}(X,A)\mid X,A\right]\right]\right\}=$$

$$\underset{Q\in\mathcal{F}}{\operatorname{argmin}}\,\|Q-Q^{\pi}\|_{2,\mu}^2,$$

as the variance term $\mathbb{E}\left[\operatorname{Var}\left[G^{\pi}(X,A)\mid X,A\right]\right]$ is not a function of Q, so it does not change the minimizer. If we could find the minimizer of $\mathbb{E}\left[|Q(X,A)-G^{\pi}(X,A)|^2\right]$, the solution would be the same as the minimizer of (34).

Recall that we encountered a similar conclusion in (3) for the supervised learning problem. This is not surprising because when with MC estimates as the target values, we are solving a supervised learning problem.

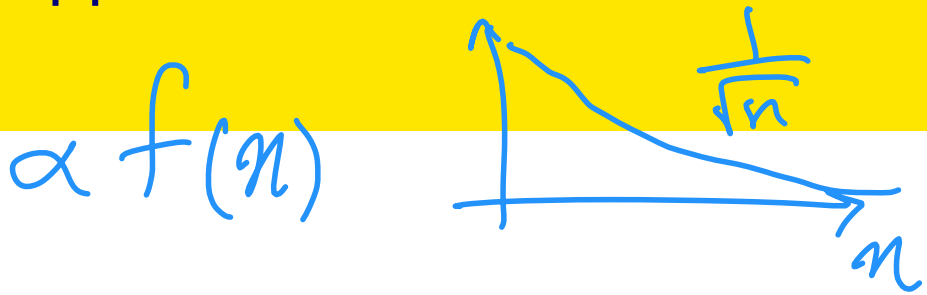
Value Function Approximation Given the Monte Carlo Estimates

- We cannot compute the expectation because we do not know μ.
- We only have samples from it.
- A common solution in ML to address this issue is to use the empirical risk minimization (ERM), which prescribes that we solve

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} |Q(X_i, A_i) - G^{\pi}(X_i, A_i)|^2 = \|Q - G^{\pi}\|_{2,\mathcal{D}_n}^2 . \tag{35}$$

- This is indeed a regression problem with the squared error loss.
- We can can add a regularizer too.

Value Function Approximation Given the Monte Carlo Estimates



Some questions:

- How close is this $\hat{Q}$ to the minimizer of $\|Q - Q^\pi\|_{2,\mu}$?
- How close is it to Q^π?
- What is the effect of the number of samples n?
- What about the effect of the function space $\mathcal{F}$?

Value Function Approximation Given the Monte Carlo Estimates – Statistical Guarantee

Assumption A1 We assume that

(a) $Z_i = (X_i, A_i)$ $(i = 1, \ldots, n)$ are i.i.d. samples from distribution $\mu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$.

(b) The reward distribution $\mathcal{R}(\cdot|x, a)$ is $R_{\max}$-bounded for any $(x, a) \in \mathcal{X} \times \mathcal{A}$.

(c) The functions in $\mathcal{F}$ are $Q_{\max} = \frac{R_{\max}}{1-\gamma}$ bounded.

(d) The function space has a finite number of members $|\mathcal{F}| < \infty$.

Value Function Approximation Given the Monte Carlo Estimates – Statistical Guarantee

Theorem (Regression)

Consider the solution $\hat{Q}$ *returned by solving* (35). *Suppose that Assumption A1 holds. For any* $\delta > 0$, *we then have*

$$\left\| \hat{Q} - Q^{\pi} \right\|_{2,\mu}^{2} \leq \inf_{Q \in \mathcal{F}} \| Q - Q^{\pi} \|_{2,\mu}^{2} + 8 Q_{max}^{2} \sqrt{\frac{2[\ln(6|\mathcal{F}|) + 2\ln(1/\delta)]}{n}},$$

with probability at least $1 - \delta$.

- Approximation Error
- Estimation Error

Approximate Value Iteration

The iteration of (6),

$$Q_{k+1} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ_k\|_{2,\mu}^2 = \int |Q(x,a) - (TQ_k)(x,a)|^2 \, \mathrm{d}\mu(x,a), \tag{36}$$

cannot be computed as

- μ is not known
- TQ_k cannot be computed as $\mathcal{P}$ is not available under the batch RL setting (33).

We can use samples though, similar to how we converted the population-level loss function (34) to the empirical one (35).

Approximate Value Iteration

If we are only given tuples in the form of (X, A, R, X'), we cannot compute

$$(T^\pi Q)(X, A) = r(X, A) + \gamma \int \mathcal{P}(\mathrm{d}x'|X, A) Q(x', \pi(x')),$$

or similar for $(T^* Q)(X, A)$.
We can, however, form an unbiased estimate of them.
We use the empirical Bellman operator applied to Q:

$$(\hat{T}^\pi Q)(X, A) = R + \gamma Q(X', \pi(X')),$$
$$(\hat{T}^* Q)(X, A) = R + \gamma \max_{a' \in \mathcal{A}} Q(X', a').$$

For any integrable Q, they satisfy

$$\mathbb{E}\left[(\hat{T} Q)(X, A)|X, A\right] = (TQ)(X, A).$$

Approximate Value Iteration

Replacing TQ_k with $\hat{T}Q_k$ does not change the optimizer.
Given any independent $Z = (X, A)$, we have

$$\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right] =$$

$$\mathbb{E}\left[\left|Q(Z) - (TQ_k)(Z) + (TQ_k)(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right] =$$

$$\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2 \mid Z\right] +$$

$$\mathbb{E}\left[\left|(TQ_k)(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right] +$$

$$2\mathbb{E}\left[(Q(Z) - (TQ_k)(Z))\left((TQ_k)(Z) - (\hat{T}Q_k)(Z)\right) \mid Z\right].$$

As $\mathbb{E}\left[(\hat{T}Q_k)(Z) \mid Z\right] = TQ_k(Z)$, the last term is zero.

Approximate Value Iteration

Conditioned on Z, the function $Q(Z) - (TQ_k)(Z)$ is not random, so $\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2 \mid Z\right] = |Q(Z) - (TQ_k)(Z)|^2$.
Therefore, we get that

$$\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right] =$$

$$|Q(Z) - (TQ_k)(Z)|^2 + \mathbb{E}\left[\left|(TQ_k)(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right] =$$

$$|Q(Z) - (TQ_k)(Z)|^2 + \mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right].$$

Approximate Value Iteration

Taking expectation over $Z \sim \mu$, we have that

$$\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right] =$$

$$\mathbb{E}\left[\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right]\right] =$$

$$\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right].$$

The term $\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2\right]$ is $\|Q - TQ_k\|_{2,\mu}^2$.

$\overset{?}{=} \operatorname{argmin}_{Q} \|Q - TQ_k\|_{2,\mu}^2$

Approximate Value Iteration

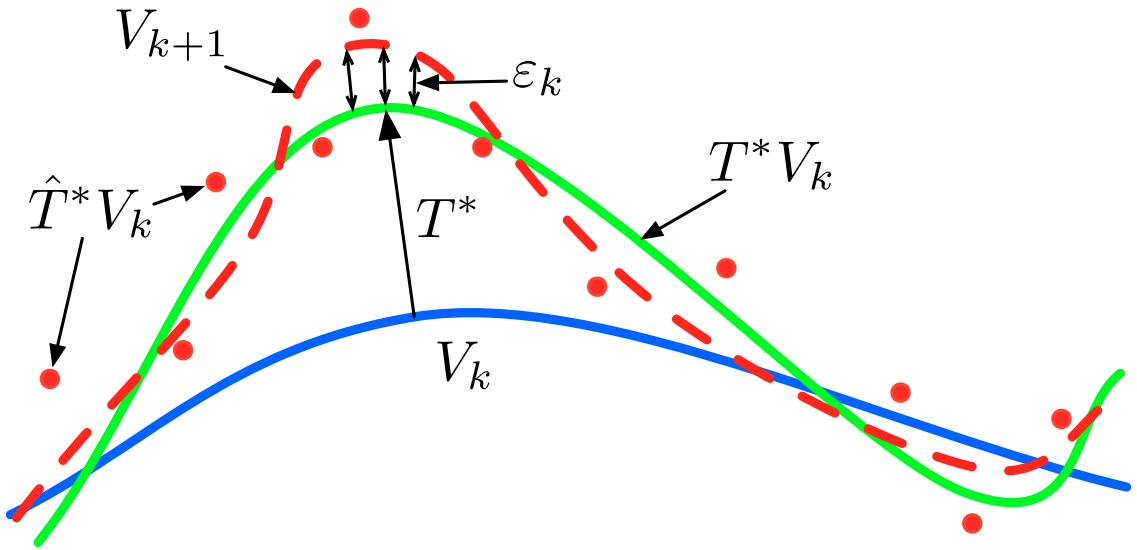
$$\operatorname*{argmin}_{Q \in \mathcal{F}} \mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right] =$$

$$\operatorname*{argmin}_{Q \in \mathcal{F}} \left\{\|Q - TQ_k\|_{2,\mu}^2 + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right]\right\} =$$

$$\operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ_k\|_{2,\mu}^2 ,$$

Instead of (36), we can minimize $\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right]$.

Approximate Value Iteration



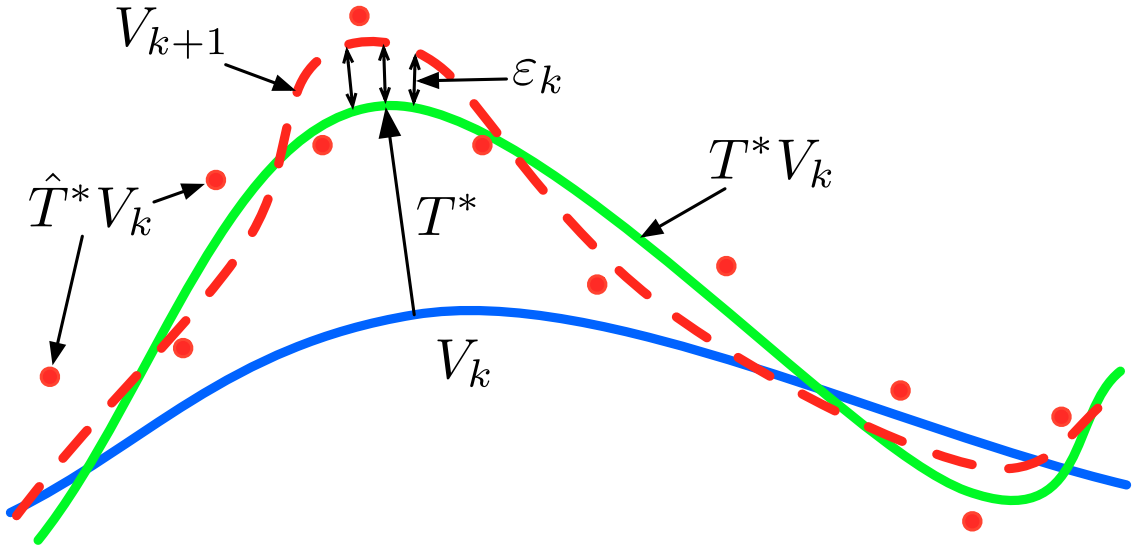
We do not have μ though.

We can use samples and form the empirical loss function. The result is the following ERM problem:

$$\hat{Q}_{k+1} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left| Q(X_i, A_i) - (\hat{T}Q_k)(X_i, A_i) \right|^2 = \left\| Q - \hat{T}Q_k \right\|_{2,\mathcal{D}_n}^2 .$$

$+\lambda J(Q)$

Approximate Value Iteration



This is the AVI procedure, also known as the Fitted Value Iteration (FVI) or Fitted Q Iteration (FQI) algorithm.
This is the basis of the Deep Q-Network (DQN) algorithm, where one uses a DNN to represent the value function space.
We can add regularizer to avoid overfitting. It is called Regularized Fitted Q Iteration (RFQI) algorithm.

Bellman Residual Minimization

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - T^\pi Q\|_{2,\mu}^2 , \tag{37}$$

Empirical version:

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left| Q(X_i, A_i) - (\hat{T}^\pi Q)(X_i, A_i) \right|^2 = \left\| Q - \hat{T}^\pi Q \right\|_{2,\mathcal{D}_n}^2 .$$

Using $\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ to convert

- integration w.r.t. μ to an integration w.r.t. μ_n
- substitute $T^\pi Q$ to its empirical version $\hat{T}^\pi Q$.

Bellman Residual Minimization

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left| Q(X_i, A_i) - (\hat{T}^\pi Q)(X_i, A_i) \right|^2 = \left\| Q - \hat{T}^\pi Q \right\|_{2,\mathcal{D}_n}^2 .$$

- Q appears in both terms inside the norm.
- As opposed to only the first term in AVI/FQI.
- This causes an issue: the minimizers of $\|Q - T^\pi Q\|_{2,\mu}^2$ and $\|Q - \hat{T}^\pi Q\|_{2,\mu}^2$ are not necessarily the same for stochastic dynamics.

Bellman Residual Minimization

To see this, we compute $\mathbb{E}\left[|Q(Z) - (\hat{T}^\pi Q)(Z)|^2 \mid Z\right]$ for any Q and $Z = (X, A)$:

$$\mathbb{E}\left[\left|Q(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] =$$

$$\mathbb{E}\left[\left|Q(Z) - (T^\pi Q)(Z) + (T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] =$$

$$\mathbb{E}\left[|Q(Z) - (T^\pi Q)(Z)|^2 \mid Z\right] +$$

$$\mathbb{E}\left[\left|(T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] +$$

$$2\mathbb{E}\left[(Q(Z) - (T^\pi Q)(Z))\left((T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right) \mid Z\right].$$

Bellman Residual Minimization

The inner product term is zero:

$$(Q(Z) - (T^\pi Q)(Z)) \left((T^\pi Q)(Z) - \mathbb{E}\left[(\hat{T}^\pi Q)(Z) \mid Z \right] \right) =$$
$$(Q(Z) - (T^\pi Q)(Z)) \left((T^\pi Q)(Z) - (T^\pi Q)(Z) \right) = 0.$$

Given Z, there is no randomness in $|Q(Z) - (T^\pi Q)(Z)|^2$:

$$\mathbb{E}\left[|Q(Z) - (T^\pi Q)(Z)|^2 \mid Z \right] = |Q(Z) - (T^\pi Q)(Z)|^2 .$$

Therefore,

$$\mathbb{E}\left[\left| Q(Z) - (\hat{T}^\pi Q)(Z) \right|^2 \mid Z \right] =$$
$$|Q(Z) - (T^\pi Q)(Z)|^2 + \mathrm{Var}\left[(\hat{T}^\pi Q)(Z) \mid Z \right].$$

Bellman Residual Minimization

$$\operatorname*{argmin}_{Q\in\mathcal{F}} \mathbb{E}\left[\left|Q(Z) - (\hat{T}Q)(Z)\right|^2\right] = \tag{38}$$

$$\operatorname*{argmin}_{Q\in\mathcal{F}} \left\{\|Q - TQ\|_{2,\mu}^2 + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q)(Z) \mid Z\right]\right]\right\} \neq$$

$$\operatorname*{argmin}_{Q\in\mathcal{F}} \|Q - TQ\|_{2,\mu}^2 .$$

- For stochastic dynamical systems, the variance term is non-zero, in general.
- For deterministic ones, it is zero.
- By replacing $T^\pi Q$ with $\hat{T}^\pi Q$ in BRM in stochastic dynamics, we obtain a solution that may not be the same as minimizer of the Bellman error within the function space $\mathcal{F}$.

Bellman Residual Minimization

Let us take a closer look at the variance term.
For simplicity, assume that the reward is deterministic, so $R_i = r(X_i, A_i)$.

$$\mathrm{Var}\left[(\hat{T}^\pi Q)(Z) \mid Z\right] =$$

$$\mathbb{E}\left[\left|(r(Z) + \gamma Q(X', \pi(X'))) - \left(r(Z) + \gamma \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x'))\right)\right|^2 \mid Z\right] =$$

$$\gamma^2 \mathbb{E}\left[\left|Q(X', \pi(X')) - \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x'))\right|^2 \mid Z\right].$$

Bellman Residual Minimization

$$\mathrm{Var}\left[(\hat{T}^{\pi}Q)(Z) \mid Z\right] = \gamma^2 \mathbb{E}\left[\left|Q(X', \pi(X')) - \int \mathcal{P}(\mathrm{d}x'|Z)Q(x', \pi(x'))\right|^2 \mid Z\right].$$

- This is the variance of Q at the next-state X'.
- Having this variance term in optimization (38) encourages finding Q that has small next-state variance.
  - If Q is constant, this term would be zero.
  - If Q is varying slowly as a function of state x (i.e., a smooth function) and the next-state distribution is localized (so not multi-modal with vastly far modes), it is going to be small.



- This induced smoothness is not desirable because it is not a smoothness that is natural to the problem, but imposed by the biased objective.
- Moreover, it is not controllable in the sense that we can change its amount by a hyperparameter.

Least Squares Temporal Difference Learning

Starting from the PBE, we got several approaches to approximate V^π.
One of them was based on solving $V = (\Pi_{\mathcal{F},\mu} T^\pi V)$ (21) with V being a linear FA with $V_N = \Phi_{N\times p} w_p$.
We showed that the solution to this equation is

$$w_{\text{TD}} = A^{-1}_{p\times p} b_{p\times 1}$$

with

$$A = \Phi^\top M(\Phi - \gamma \mathcal{P}^\pi \Phi),$$
$$b = \Phi^\top M r^\pi.$$

We need to use data $\mathcal{D}_n$ in order to estimate these.

Least Squares Temporal Difference Learning

Expand A and b in terms of summation.
As M is a diagonal matrix, we have

$$A_{ij} = \left[\Phi^\top M(\Phi - \gamma \mathcal{P}^\pi \Phi)\right]_{ij} = \sum_{m=1}^{N} \Phi_{im}^\top \mu(m) \left(\Phi - \gamma \mathcal{P}^\pi \Phi\right)_{mj},$$

$$b_i = \sum_{m=1}^{N} \Phi_{im}^\top \mu(m) r_m^\pi$$

Or expanded more explicitly in terms of state x and next-state x',

$$A = \sum_{x \in \mathcal{X}} \mu(x)\phi(x) \left(\phi(x) - \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x)\phi(x')\right)^\top,$$

$$b = \sum_{x \in \mathcal{X}} \mu(x)\phi(x) r^\pi(x).$$

Least Squares Temporal Difference Learning

Given $\mathcal{D}_n = \{(X_i, R_i, X_i')\}_{i=1}^n$ with $X_i \sim \mu \in \mathcal{M}(\mathcal{X})$, and $X_i' \sim \mathcal{P}^\pi(\cdot|X_i)$ and $R_i \sim \mathcal{R}^\pi(\cdot|X_i)$, we define the empirical estimates $\hat{A}_n$ and $\hat{b}_n$ as

$$\hat{A}_n = \frac{1}{n}\sum_{i=1}^n \phi(X_i)\left(\phi(X_i) - \gamma\phi(X_i')\right)^\top,$$

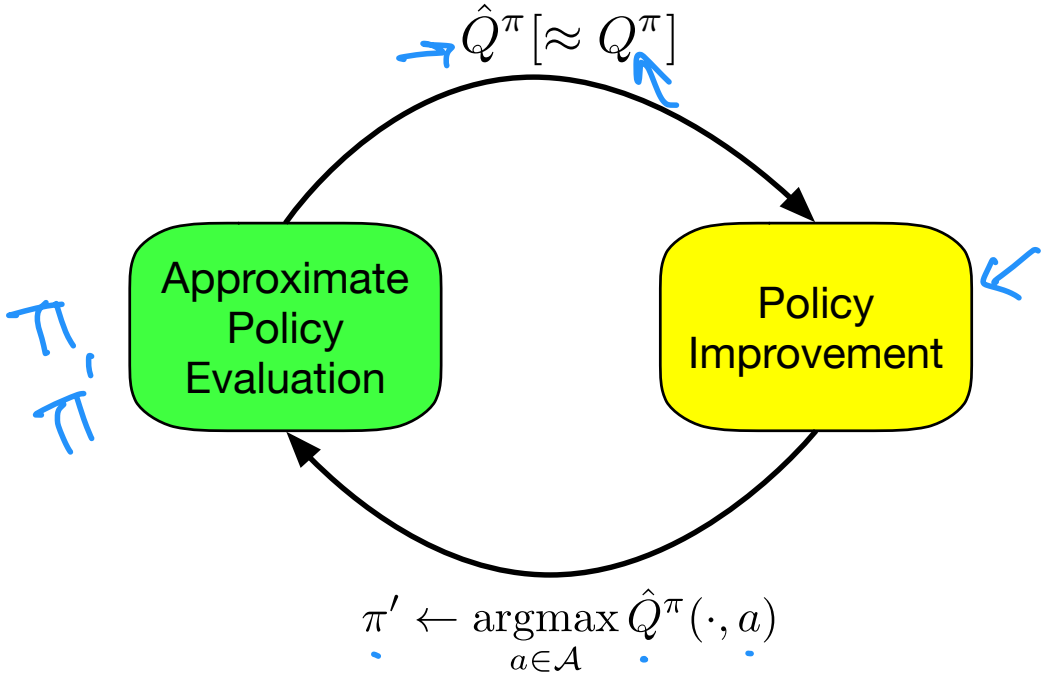
$$\hat{b}_n = \frac{1}{n}\sum_{i=1}^n \phi(X_i) R_i.$$

- These are unbiased estimates of A and b (Exercise: Prove it!)
- If X_is are independent (or from a Markov chain with some regularity conditions), by the LLN, they converge to A and b.

Least Squares Temporal Difference Learning

We may use LSTD to estimate the action-value function Q^π too.
See the Foundations of RL book [Farahmand, 2025] for details.

Least Squares Policy Iteration (LSPI)



- We can use LSTD to define an approximate PI (API) procedure to obtain a close to optimal policy.
- This is a policy iteration algorithm that uses LSTD to evaluate a policy.
- It is approximate because of the use of a function approximation and a finite number of data points.

Least Squares Policy Iteration (LSPI)

Require: $\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ and initial policy π_1.

1: **for** $k = 1, 2, \ldots, K$ **do**
2: $\quad \hat{Q}^{\pi_k} \leftarrow \mathrm{LSTD}(\mathcal{D}_n, \pi_k)$ ▷ Policy Evaluation
3: $\quad \pi_{k+1} \leftarrow \pi_g(\hat{Q}^{\pi_k})$ ▷ Policy Improvement
4: **end for**
5: **return** $\hat{Q}^{\pi_K}$ and π_{K+1}.

Least Squares Policy Iteration (LSPI)

- We may also collect more data during LSPI.
  - As we obtain a new policy, we can follow it to collect new data points.
- LSTD (for action-value function Q) is an off-policy algorithm because it can evaluate a policy π that is different from the one collecting data.
- LSTD and LSPI are considered as sample efficient algorithms.
- They are, however, not computationally cheap.
  - The matrix inversion $\hat{A}_{p\times p}$ is $O(p^3)$, if computed naively.
  - If we want to perform it in an online fashion, as new samples arrive, the computational cost can be high: $O(np^3)$.
  - We may use Sherman-Morrison formula to compute $\hat{A}_n^{-1}$ incrementally based on the previous inverted matrix $\hat{A}_{n-1}^{-1}$.

Online RL Methods

Online RL Methods with Function Approximation

In the online setting, the agent updates the value function as it interacts with the environment. We can use the update rules derived before (2) in order to design a SA procedure.

Online Method #1

We consider the weight update rule (30):

$$w_{k+1} \leftarrow (1-\alpha)w_k + \alpha(\Phi^\top D^\pi \Phi)^{-1}\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right].$$

In order to convert this to a SA procedure, we need to empirically estimate

- $\Phi^\top D^\pi \Phi$
- $\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right]$.

Online Method #1

We have

$$(\Phi^\top D^\pi \Phi)_{p\times p} = \sum_{x\in\mathcal{X}} \rho^\pi(x)\phi(x)\phi^\top(x)$$
$$= \mathbb{E}\left[\phi(X)\phi^\top(X)\right], \qquad X \sim \rho^\pi.$$

If we have t data points $X_1, \dots X_t$ with $X_i \sim \rho^\pi$, the stationary distribution of π, we can estimate it by a matrix $\hat{A}_t$

$$\hat{A}_t = \frac{1}{t}\sum_{i=1}^{t} \phi(X_i)\phi^\top(X_i).$$

This matrix is an unbiased estimate of $(\Phi^\top D^\pi \Phi)$, and converges to it under usual conditions of LLN.

Online Method #1

We also have

$$\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right] =$$

$$\sum_{x \in \mathcal{X}} \rho^\pi(x)\phi(x) \left(r^\pi(x) + \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x)\phi^\top(x')w_k \right). \tag{39}$$

If $X_t \sim \rho^\pi$, $X'_t \sim \mathcal{P}^\pi(\cdot|X_t)$, and $R_t \sim \mathcal{R}^\pi(\cdot|X_t)$, the r.v.

$$\phi(X_t)\left(R_t + \gamma \phi^\top(X'_t)w_t\right)$$

is an unbiased estimate of (39).

Online Method #1

Stochastic Approximation: at each time step t, after observing X_t, R_t, X'_t, update the weight w_t to w_{t+1} by

$$w_{t+1} \leftarrow (1-\alpha_t)w_t + \alpha_t \hat{A}_t^{-1}\phi(X_t)\left(R_t + \gamma\phi^\top(X'_t)w_t\right) \tag{40}$$

with

$$\begin{aligned}\hat{A}_t &= \frac{1}{t}\left[(t-1)\hat{A}_{t-1} + \phi(X_t)\phi^\top(X_t)\right] \\ &= \left(1-\frac{1}{t}\right)\hat{A}_{t-1} + \frac{1}{t}\phi(X_t)\phi^\top(X_t).\end{aligned}$$

Online Method #1

The inversion of $\hat{A}_t$ is expensive, if done naively (we need t to be large enough for $\hat{A}_t$ to be full rank and hence invertible). Consider the summation $\hat{S}_t = \sum_{i=1}^{t} \phi(X_i)\phi^\top(X_i)$, so $\hat{A}_t^{-1} = t\hat{S}_t^{-1}$. We can now use Sherman-Morrison formula to incrementally update $\hat{S}_t^{-1}$:

$$(\hat{S}_t)^{-1} = \hat{S}_{t-1}^{-1} - \frac{\hat{S}_{t-1}^{-1}\phi(X_t)\phi^\top(X_t)\hat{S}_{t-1}^{-1}}{1 + \phi^\top(X_t)\hat{S}_{t-1}^{-1}\phi(X_t)}.$$

Remark (Sherman-Morrison formula)

For an invertible matrix $A_{d\times d}$ and vectors $u, v \in \mathbb{R}^d$, the matrix $A + uv^\top$ is invertible if and only if $1 + v^\top A^{-1}u \neq 0$. And if it is invertible, we can compute it as

$$\left(A + uv^\top\right)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.$$

Online Method #1 – Computational Complexity

The update rule becomes

$$w_{t+1} \leftarrow (1-\alpha_t)w_t + \alpha_t t \hat{S}_t^{-1}\phi(X_t)\left(R_t + \gamma\phi^\top(X_t')w_t\right) \tag{41}$$

with

$$(\hat{S}_t)^{-1} = \hat{S}_{t-1}^{-1} - \frac{\hat{S}_{t-1}^{-1}\phi(X_t)\phi^\top(X_t)\hat{S}_{t-1}^{-1}}{1+\phi^\top(X_t)\hat{S}_{t-1}^{-1}\phi(X_t)}.$$

- This requires a matrix-vector multiplication and is $O(p^2)$.
- The per-sample computational cost of (40) (and (41)) is then $O(p^2)$.
- This is significantly higher than the $O(1)$ computational cost of the TD update for a problem with finite state-action spaces for which the value function can be represented exactly in a lookup table.

Online Method #1 – Computational Complexity

- This comparison may not be completely fair:
  - The computational cost of evaluating $V(x)$ at any x for a finite state problem with an exact representation was $O(1)$ itself.
  - The computational cost of evaluating the value function with a linear FA with p features (i.e., $V(x; w) = \phi^\top(x)w$) is $O(p)$.
- A better baseline is to compare the cost of update per time step with the cost of computation of V for a single state:

$$\frac{\text{cost of update per sample}}{\text{cost of computing the value of a single state}}.$$

- For TD with a finite state(-action) space with the exact representation, the ratio is $O(1)$.
- For the method (40), the ratio is $O(p)$.
- More graceful dependence on p, but still scales linearly with the number of features.

Online Method #2: TD Learning

We can have a better computational cost using the other update rule (32). The population version:

$$w_{k+1} \leftarrow w_k + \alpha \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right].$$

Online Method #2: TD Learning

$$w_{k+1} \leftarrow w_k + \alpha \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right].$$

If $X_t \sim \rho^\pi$, $X'_t \sim \mathcal{P}^\pi(\cdot|X_t)$, and $R_t \sim \mathcal{R}^\pi(\cdot|X_t)$, we use the r.v.

$$\phi(X_t) \underbrace{\left(R_t + \gamma \phi^\top(X'_t) w_t - \phi^\top(X_t) w_t \right)}_{\delta_t} = \phi(X_t)\delta_t,$$

with the TD error

$$\delta_t = R_t + \gamma \phi^\top(X'_t) w_t - \phi^\top(X_t) w_t.$$

This is an unbiased estimate of $\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right]$.

Online Method #2: TD Learning

The SA update rule would be

$$w_{t+1} \leftarrow w_t + \alpha_t \phi(X_t)\delta_t. \tag{42}$$

This is the TD Learning with linear FA.

Online Method #2: TD Learning

- The population version of this update rule under $X \sim \rho^\pi$ converges (see FRL).
- We do not show it for the SA version, but we might suspect that it does because it follows a noise contaminated version of a stable/convergent dynamical system.
- With proper choice of the step size sequence (α_t), we can expect convergence.
- This is indeed true, as shown by Tsitsiklis and Van Roy [1997].
- This convergence holds when $X_t \sim \rho^\pi$, the stationary distribution of π.
- If its distribution is not the same, the TD with linear FA might diverge.
- This is in contrast with the TD for finite state problems where the conditions of convergence were much easier and we did not have divergence.

Online Method #2: TD Learning

The same method works for learning an action-value function Q^π of policy π using an approximation

$$Q(x,a) = Q(x,a;w) = \phi(x,a)^\top w.$$

For $X_t \sim \rho^\pi$, $A_t = \pi(X_t)$, $X'_t \sim \mathcal{P}(\cdot|X_t, A_t)$, and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$, we can update the weights as

$$w_{t+1} \leftarrow w_t + \alpha_t \phi(X_t, A_t)\delta_t, \tag{43}$$

with the TD error

$$\delta_t = R_t + \gamma \phi(X'_t, \pi(X'_t))^\top w_t - \phi(X_t, A_t)^\top w_t. \tag{44}$$

Online Method #2: TD Learning

- We may use a similar procedure for the control problem and define SARSA-like and Q-Learning-like algorithms with linear FA.
- For SARSA, the update uses the tuple $(X_t, A_t, R_t, X'_t, A'_t)$ with $A_t \sim \pi(\cdot|X_t)$ and $A'_t \sim \pi(\cdot|X'_t)$, and π being a policy that is close to being greedy w.r.t. Q_t, e.g., an ε-greedy policy $\pi_\varepsilon(Q_t)$.
- The update would be the same with the difference that the TD error would be

$$\delta_t = R_t + \gamma \phi(X'_t, A'_t)^\top w_t - \phi(X_t, A_t)^\top w_t.$$

Online Method #2: TD Learning

We may also form a Q-Learning-like algorithm by having

$$\delta_t = R_t + \gamma \max_{a' \in \mathcal{A}} \phi(X'_t, a')^\top w_t - \phi(X_t, A_t)^\top w_t$$

- Even though the agent may be following a policy π and have samples $X_t \sim \rho^\pi$ and $A_t \sim \pi(\cdot|X_t)$ (or similar for the deterministic policy), the policy being evaluated is the greedy policy $\pi_g(\cdot; Q_t)$.
- The evaluated policy may not be the same as π.
- This is an off-policy sampling scenario.
- The convergence guarantee for TD with linear FA, shown by Tsitsiklis and Van Roy [1997], does not hold here.
- In fact, Q-Learning with linear FA might diverge.

Semi-Gradient Viewpoint

- We motivated the TD method with linear FA by starting from $V = \Pi_{\mathcal{F},\rho^\pi} T^\pi V$ with $V = \Phi w$, and devised an iterative SA procedure for its computation.
- One may also see it as an SGD-like procedure, with some modifications, as we explain here.
- It is the approach followed by Sutton and Barto [2018].

Semi-Gradient Viewpoint

Suppose that we know the true value function V^π, and we want to find an approximation $\hat{V}$, parameterized by w. The population loss is

$$\hat{V} \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \frac{1}{2} \|V^\pi - V\|_{2,\mu}^2 . \tag{45}$$

When $\hat{V}$ is parameterized by w, the optimization problem can be written as

$$\hat{w} \leftarrow \operatorname*{argmin}_{w} \frac{1}{2} \left\|V^\pi - \hat{V}(w)\right\|_{2,\mu}^2 . \tag{46}$$

Semi-Gradient Viewpoint

Using samples $X_t \sim \mu$, we can define an SGD procedure that updates w_t as follows:

$$\begin{aligned} w_{t+1} \leftarrow & w_t - \alpha_t \nabla_w \left[\frac{1}{2} \left| V^\pi(X_t) - \hat{V}(X_t; w_t) \right|^2 \right] \\ = & w_t + \alpha_t \left(V^\pi(X_t) - \hat{V}(X_t; w_t) \right) \nabla_w \hat{V}(X_t; w_t). \end{aligned}$$

If the step size α_t is selected properly and some other conditions are satisfied, the SGD converges to the stationary point of the objective of (45).

If we use a linear FA to represent $\hat{V}$, the objective would be convex, so w_t converges to the global minimum of the objective.

$V^\pi(X_t)$ acts as the target, in the supervised learning sense.

Semi-Gradient Viewpoint

When we do not know V^π, we may use a bootstrapped estimate instead:

$$(\hat{T}^\pi V_t)(X_t) = R_t + \gamma V_t(X'_t) = R_t + \gamma \hat{V}(X'_t; w_t).$$

With this substitution, the update rule would be

$$w_{t+1} \leftarrow w_t + \alpha_t \left(R_t + \gamma \hat{V}(X'_t; w_t) - \hat{V}(X_t; w_t) \right) \nabla_w \hat{V}(X_t; w_t).$$

$\simeq V^\pi(X_t)$

Semi-Gradient Viewpoint

For linear FA, we have $\hat{V}(x; w) = \phi^\top(x)w$, and we get the update rule

$$\begin{aligned} w_{t+1} \leftarrow & w_t + \alpha_t \left(R_t + \gamma \hat{V}(X'_t; w_t) - \hat{V}(X_t; w_t) \right) \phi(X_t) \\ = & w_t + \alpha_t \delta_t \phi(X_t). \end{aligned}$$

This is the same update rule that we had before for TD with linear FA (42).

Remark

The substitution of $V^\pi(X_t)$ with $(\hat{T}^\pi V_t)(X_t)$ does not follow from the SGD of any loss function.
The TD update is not a true Stochastic Gradient Descent (SGD) update.
We call it a semi-gradient update.

Semi-Gradient Viewpoint

One may wonder why we do not perform SGD on

$$\frac{1}{2}\left|\hat{V}(X_t;w_t)-(\hat{T}^\pi\hat{V}(w_t))(X_t)\right|^2$$

Certainly, we can write

$$\begin{aligned}
&w_{t+1}\leftarrow w_t-\alpha_t\nabla_w\left[\frac{1}{2}\left|\hat{V}(X_t;w_t)-(\hat{T}^\pi\hat{V}(w_t))(X_t)\right|^2\right]\\
=&w_t-\alpha_t\left(\hat{V}(X_t;w_t)-(\hat{T}^\pi\hat{V}(w_t))(X_t)\right)\left(\nabla_w\hat{V}(X_t;w_t)-\nabla_w(\hat{T}^\pi\hat{V}(w_t))(X_t)\right)\\
=&w_t-\alpha_t\left(\hat{V}(X_t;w_t)-(\hat{T}^\pi\hat{V}(w_t))(X_t)\right)\left(\nabla_w\hat{V}(X_t;w_t)-\gamma\nabla_w\hat{V}(X_t';w_t)\right)
\end{aligned}$$

Semi-Gradient Viewpoint

With linear FA, this becomes

$$\begin{aligned} w_{t+1} \leftarrow & w_t - \alpha_t \left(\phi(X_t)^\top w_t - (R_t + \gamma \phi(X_t')^\top w_t \right) \left(\phi(X_t) - \gamma \phi(X_t') \right) \\ = & w_t + \alpha_t \delta_t \cdot \left(\phi(X_t) - \gamma \phi(X_t') \right) \end{aligned}$$

with $\delta_t = R_t + \gamma \max_{a' \in \mathcal{A}} \phi(X_t', a')^\top w_t - \phi(X_t, A_t)^\top w_t$ (44).

- This is similar to the TD update, with the difference that instead of $\phi(X_t)$, we have $\phi(X_t) - \gamma \phi(X_t')$.
- The issue, however, is that this empirical loss function $\frac{1}{2} |\hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t)|^2$ is biased, as explained in Section 3.
- Minimizing it does not lead to the minimizer of the Bellman error.

Gradient Temporal Difference Learning

We can also define an online algorithm to minimize MSPBE:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - \Pi_{\mathcal{F},\mu} T^{\pi} V\|_{2,\mu}^2 .$$

Summary

- Function approximation is needed for large problems
- Several approaches for approximating the value function
  - Direct estimate of V^π
  - Bellman Error
  - Projected Bellman Error
- Batch RL methods
  - AVI/FQI
  - LSTD
- Online RL methods
  - TD Learning with linear FA
  - TD-like update
- Next: Policy Search methods

References

Amir-massoud Farahmand. *Foundations of Reinforcement Learning (draft)*. 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.