- Lecture 15 Notes: Value Function Geometry

- Notes Start Slide 7, 20 minute mark

$B^{\pi}(V)$ is a linear operator in vector space $\mathbb{R}^n$

↳ Conceptualize B^{π} as an $n \times n$ matrix

$\therefore \; B^{\pi} V^{\pi} = V^{\pi}$

↳ V^{π} is a fixed point. By fixed point theorem, you can start with any function V and you will converge on fixed point V^{π} after some number of iterations

↳ Monte Carlo also converges to V^{π}, but much slower than DP.

- Projection Operator Π_{Φ}: Takes any vector in n dimensional space and drops it perpendicular to the subspace of the function approx.

↳ Define "distance" $d(V_1, V_2)$ between VF vectors V_1, V_2

↳ Weighted average of squared differences, weighted by probabilities

$$d(V_1, V_2) = \sum_{i=1}^{n} \mu_{\pi}(s_i) \cdot (V_1(s_i) - V_2(s_i))^2 = (V_1 - V_2)^T \cdot D \cdot (V_1 - V_2)$$

↳ Weighted linear regression

↳ Review slide 8 discussion

- We want to find the closest approximate VF to true VF :

① $W_\pi = \arg\min \; d(V_\pi, V_w)$

②

③

④ Projected Bellman Error (PBE) minimizing

↪ Each operator is $\Pi_\phi \cdot B_\pi$

Projects onto plane

Throws out of plane

↪ Also a contraction that converges on fixed point

- Slide 11 : Geometric view of RL

- Normal TD does not converge as closely as MC to real VF, but it is more efficient

- The right quantity to minimize for RL is projected Bellman error

- See slide 12 for optimal W_{BE} which minimizes Bellman error using linear function approx

↪ Model Free approach (slide 13) applies Sherman Morrison Inverse

↪ Batch RL way of minimizing Bellman Error

↪ Only works for small number of features

↪ With many features, use residual gradient algorithm

↪ $\Delta W = -\frac{1}{2} \alpha \cdot \nabla_w \left(\mathbb{E}_\pi(\delta) \right)^2$

TD Error δ

↪ BE ⇔ Expected TD Error δ when following policy π

↪ Disadvantage : Product of 2 conditional expectations requires 2 independent samples of s'.

→ Works for Simulation, not in real world

- Bellman error did not produce strong results ∴ People switched to projected Bellman Error

- Without probabilities, do RL. When MC is too slow, use Bootstrap RL w/ function approx.

  → Can use BE minimizing weights, Projected BE minimizing weights, and TD Error minimizing

- Minimizing TD Error : W_{TDE}

  → Expected Square of TD error δ when following policy π

  → Minimizes robustly, but converges to point far from optimal

- Minimizing Projected Bellman Error

  → Math on slide 16

  → There exists a fixed point

$$\underbrace{\Pi_\phi \; B^\pi}_{\text{Product of matricies}}$$

← Slide 16 shows product converging to fixed point.

→ Model free approach removes the need for probabilities by incorporating all probability terms into a matrix A

→ This is Least Squares Temporal Difference Method

→ LSTD is the best solution you can get with bootstrapping methods

→ Alternatively: We can use Semi-Gradient TD descent with updates:

→ Lands you at the same spot as LSTD

$$\Delta w = \alpha \left(r + \gamma \cdot \phi(s')^T \cdot w - \phi(s)^T \cdot w \right) \cdot \phi(s)$$

- Projected Bellman Error works with semi-gradient descent if operating with linear, on-policy funct. approx.

- If you operate off-policy, you run into deadly triad.
  - → Because semi-gradient is an approximation

- Slides 18 - 20 evaluate W_{PBE}

- <u>Slide 20 :</u> Cascade learning. W depends on θ, θ converges much faster than W and pulls W with it.