Off-policy Methods with Approximation

- Off-policy: learn a value function for a target policy given data due to a different behaviour policy
- Challenges:
    - Target of the update (importance sampling)
    - Distribution of the update (importance sampling or true gradient)

Semi-gradient Methods

- Per-step importance sampling ratio: $\rho_t \doteq \rho_{t:t} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}.$
- Semi-gradient off-policy TD(0): $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\rho_t\delta_t\nabla\hat{v}(S_t,\mathbf{w}_t),$

$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}_t) - \hat{v}(S_t,\mathbf{w}_t),$$

- See book for Expected Sarsa, n-step methods, tree-backup
- These methods may be unstable and diverge

The Deadly Triad

- Danger of instability and divergence arises when we combine FA, bootstrapping and off-policy training
  - Occurs even in the prediction case and when there is no environment uncertainties (e.g. DP)
- Instability can be avoided by losing one of the three:
  - FA: required for large state-spaces and generalization
  - Bootstrapping: MC but loss of computational and data efficiency
  - Off-policy: Sarsa rather than Q-learning but off-policy methods free behavior from the target policy (required for parallel learning to learn many target policies from one stream of experience)

Linear Value-function Geometry

- Consider 3 states and 2 parameters of linear weight vector → value function approximation is a plane in 3D space
- The true value function is in the larger space and can be projected down into the subspace to its best approximation (VE, BE, PBE, TDE)
- Distance between value functions: $$\|v\|_\mu^2 \doteq \sum_{s\in\mathcal{S}} \mu(s)v(s)^2.$$
- Mean square value error $$\overline{\mathrm{VE}}(\mathbf{w}) \doteq \sum_{s\in\mathcal{S}} \mu(s)\Big[v_\pi(s) - \hat{v}(s,\mathbf{w})\Big]^2. \;=\; |v_{\mathbf{w}} - v_\pi\|_\mu^2.$$
- Projection operator Π: $$\Pi v \doteq v_{\mathbf{w}} \quad \text{where} \quad \mathbf{w} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \|v - v_{\mathbf{w}}\|_\mu^2 .$$

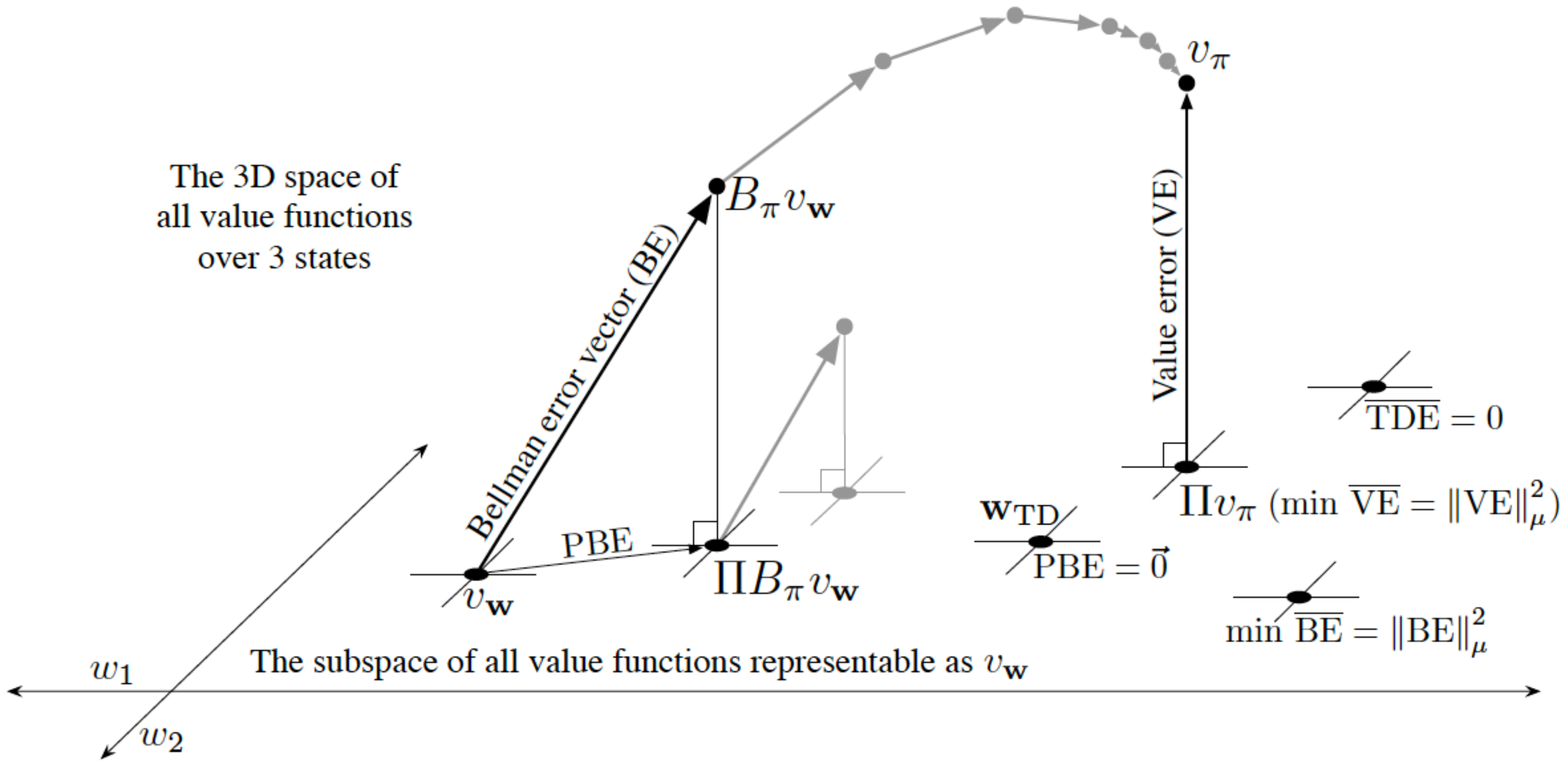

The 3D space of all value functions over 3 states
v_π
$B_\pi v_\mathbf{w}$
Bellman error vector (BE)
Value error (VE)
$\overline{\mathrm{TDE}} = 0$
Πv_π (min $\overline{\mathrm{VE}} = \|\mathrm{VE}\|_\mu^2$)
$\mathbf{w}_{\mathrm{TD}}$
PBE
$\mathrm{PBE} = \vec{0}$
$v_\mathbf{w}$
$\Pi B_\pi v_\mathbf{w}$
min $\overline{\mathrm{BE}} = \|\mathrm{BE}\|_\mu^2$
w_1
The subspace of all value functions representable as $v_\mathbf{w}$
w_2

Figure 11.3: The geometry of linear value-function approximation. Shown is the three-dimensional space of all value functions over three states, while shown as a plane is the subspace of all value functions representable by a linear function approximator with parameter $\mathbf{w} = (w_1, w_2)^\top$. The true value function v_π is in the larger space and can be projected down (into the subspace, using a projection operator Π) to its best approximation in the value error (VE) sense. The best approximators in the Bellman error (BE), projected Bellman error (PBE), and temporal difference error (TDE) senses are all potentially different and are shown in the lower right. (VE, BE, and PBE are all treated as the corresponding vectors in this figure.) The Bellman operator takes a value function in the plane to one outside, which can then be projected back. If you iteratively applied the Bellman operator outside the space (shown in gray above) you would reach the true value function, as in conventional dynamic programming. If instead you kept projecting back into the subspace at each step, as in the lower step shown in gray, then the fixed point would be the point of vector-zero PBE.

Objectives

- Mean squared Bellman error is the expectation of the TD error and cannot be zero

$$\overline{\mathrm{BE}}(\mathbf{w}) = \left\|\bar{\delta}_{\mathbf{w}}\right\|_{\mu}^{2}. \qquad \bar{\delta}_{\mathbf{w}}(s) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t) \mid S_t = s, A_t \sim \pi\right],$$

- Mean square project Bellman error $\overline{\mathrm{PBE}}(\mathbf{w}) = \left\|\Pi\bar{\delta}_{\mathbf{w}}\right\|_{\mu}^{2}.$
- Mean squared TD error $\delta_t = R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}_t) - \hat{v}(S_t,\mathbf{w}_t).$

$$\begin{aligned}
\overline{\mathrm{TDE}}(\mathbf{w}) &= \sum_{s\in\mathcal{S}} \mu(s)\mathbb{E}\left[\delta_t^2 \mid S_t = s, A_t \sim \pi\right] \\
&= \sum_{s\in\mathcal{S}} \mu(s)\mathbb{E}\left[\rho_t\delta_t^2 \mid S_t = s, A_t \sim b\right] \\
&= \mathbb{E}_b\left[\rho_t\delta_t^2\right]. \qquad \text{(if } \mu \text{ is the distribution encountered under } b\text{)}
\end{aligned}$$

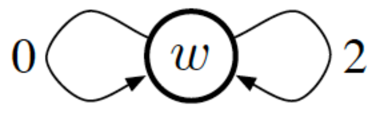
Gradient Descent in the Bellman Error

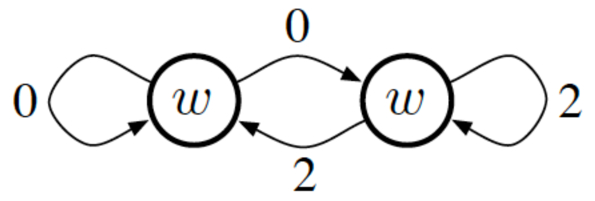
- SGD updates always go downhill (in expectation) in the objective
  → typically stable with excellent convergence properties (MC)
- Semi-gradient may diverse under off-policy
- Possible objective functions for SGD: TD error (naïve residual-gradient algorithm), Bellman error (residual-gradient algorithm)
  - MS(TD)E update:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha\nabla(\rho_t\delta_t^2)\\ &= \mathbf{w}_t - \alpha\rho_t\delta_t\nabla\delta_t\\ &= \mathbf{w}_t + \alpha\rho_t\delta_t\big(\nabla\hat{v}(S_t,\mathbf{w}_t) - \gamma\nabla\hat{v}(S_{t+1},\mathbf{w}_t)\big),\end{aligned}$$

The Bellman Error is Not Learnable

- Learnable from an infinite amount of experiential data (as opposed to it's common use of efficiently learnable i.e. poly samples vs exp)
- Cannot tell MDPs below apart from observed trajectories value error objective is not learnable (but the parameter that optimizes w* is)





- Mean square return error is learnable (i.e. Monte Carlo objective)
- Bellman error is not learnable from data and the optimal parameter vector is not a function of the data whilst PBE and TDE are learnable
- BE limited to model-based as must have access to the MDP

Monte Carlo objectives

$$\begin{tikzcd}
 & \text{Data distribution} \arrow[dr] & \\
\text{MDP}_1 \arrow[ur] \arrow[d] & \text{MDP}_2 \arrow[u] \arrow[d] & \overline{\text{RE}} \arrow[ddl] \\
\overline{\text{VE}}_1 \arrow[dr] & \overline{\text{VE}}_2 \arrow[d] & \\
 & \mathbf{w}^* &
\end{tikzcd}$$

Bootstrapping objectives

$$\begin{tikzcd}
 & \text{Data distribution} \arrow[dr] \arrow[drr] & & \\
\text{MDP}_1 \arrow[ur] \arrow[d] & \text{MDP}_2 \arrow[u] \arrow[d] & \overline{\text{PBE}} \arrow[d] & \overline{\text{TDE}} \arrow[d] \\
\overline{\text{BE}}_1 \arrow[d] & \overline{\text{BE}}_2 \arrow[d] & \mathbf{w}_3^* & \mathbf{w}_4^* \\
\mathbf{w}_1^* & \mathbf{w}_2^* & &
\end{tikzcd}$$

Figure 11.4: Causal relationships among the data distribution, MDPs, and various objectives. **Left, Monte Carlo objectives:** Two different MDPs can produce the same data distribution yet also produce different $\overline{\text{VE}}$s, proving that the $\overline{\text{VE}}$ objective cannot be determined from data and is not learnable. However, all such $\overline{\text{VE}}$s must have the same optimal parameter vector, $\mathbf{w}^*$! Moreover, this same $\mathbf{w}^*$ can be determined from another objective, the $\overline{\text{RE}}$, which *is* uniquely determined from the data distribution. Thus $\mathbf{w}^*$ and the $\overline{\text{RE}}$ are learnable even though the $\overline{\text{VE}}$s are not. **Right, Bootstrapping objectives:** Two different MDPs can produce the same data distribution yet also produce different $\overline{\text{BE}}$s *and* have different minimizing parameter vectors; these are not learnable from the data distribution. The $\overline{\text{PBE}}$ and $\overline{\text{TDE}}$ objectives and their (different) minima can be directly determined from data and thus are learnable.

Gradient-TD Methods

- For minimising PBE and as true SGD then robust convergence properties even under off-policy training and nonlinear FA
  - For linear case then exact solution at TD fixed point where PBE is zero
- See book for GTD2 and TDC algorithms
- Gradient-TD methods are currently the most well understood and widely used stable off-policy methods.

Emphatic-TD Methods

- Instead of IS (reweighting state transitions), reweight the states, emphasizing some and de-emphasizing others, so as to return the distribution of updates to the on-policy distribution
- Emphatic-TD algorithm
  - I = interest
  - M = emphasis

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t),$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha M_t \rho_t \delta_t \nabla \hat{v}(S_t, \mathbf{w}_t),$$

$$M_t = \gamma \rho_{t-1} M_{t-1} + I_t,$$

Reducing Variance

- Off-policy learning is inherently of greater variance than on-policy learning - if you receive data less closely related to a policy, you should expect to learn less about the policy's values.
- Off-policy enables generalization to this vast number of related-but-not-identical policies
- Importance sampling often involves products of policy ratios. The ratios are always one in expectation, but their actual values may be very high or as low as zero.
- Small step-size parameter = very slow learning
- Alternative: weighted IS, Tree Backup algorithm