# Value Function Geometry and Gradient TD

Ashwin Rao

ICME, Stanford University

February 18, 2019

# Overview

# Motivation for understanding Value Function Geometry

- Helps us better understand transformations of Value Functions (VFs)
- Across the various DP and RL algorithms
- Particularly helps when VFs are approximated, esp. with linear approx
- Provides insights into stability and convergence
- Particularly when dealing with the "Deadly Triad"
- Deadly Triad := [Bootstrapping, Func Approx, Off-Policy]
- **Leads us to Gradient TD**

# Notation

- Assume state space $\mathcal{S}$ consists of $n$ states: $\{s_1, s_2, \ldots, s_n\}$
- Action space $\mathcal{A}$ consisting of finite number of actions
- This exposition extends easily to continuous state/action spaces too
- This exposition is for a fixed (often stochastic) policy denoted $\pi(a|s)$
- VF for a policy $\pi$ is denoted as $\mathbf{v}_\pi : \mathcal{S} \rightarrow \mathbb{R}$
- $m$ feature functions $\phi_1, \phi_2, \ldots, \phi_m : \mathcal{S} \rightarrow \mathbb{R}$
- Feature vector for a state $s \in \mathcal{S}$ denoted as $\phi(s) \in \mathbb{R}^m$
- For linear function approximation of VF with weights $\mathbf{w} = (w_1, w_2, \ldots, w_m)$, VF $\mathbf{v}_\mathbf{w} : \mathcal{S} \rightarrow \mathbb{R}$ is defined as:

$$\mathbf{v}_\mathbf{w}(s) = \mathbf{w}^T \cdot \phi(s) = \sum_{j=1}^{m} w_j \cdot \phi_j(s) \text{ for any } s \in \mathcal{S}$$

- $\mu_\pi : \mathcal{S} \rightarrow [0, 1]$ denotes the states' probability distribution under $\pi$

# VF Geometry and VF Linear Approximations

- Consider $n$-dim space $\mathbb{R}^n$, with each dim corresponding to a state in $\mathcal{S}$
- Think of a VF (typically denoted $\mathbf{v}$): $\mathcal{S} \rightarrow \mathbb{R}$ as a vector in this space
- Each dimension's coordinate is the VF for that dimension's state
- Coordinates of vector $\mathbf{v}_\pi$ for policy $\pi$ are: $[\mathbf{v}_\pi(s_1), \mathbf{v}_\pi(s_2), \ldots, \mathbf{v}_\pi(s_n)]$
- Consider $m$ vectors where $j^{th}$ vector is: $[\phi_j(s_1), \phi_j(s_2), \ldots, \phi_j(s_n)]$
- These $m$ vectors are the $m$ columns of $n \times m$ matrix $\mathbf{\Phi} = [\phi_j(s_i)]$
- Their span represents an $m$-dim subspace within this $n$-dim space
- Spanned by the set of all $\mathbf{w} = [w_1, w_2, \ldots, w_m] \in \mathbb{R}^m$
- Vector $\mathbf{v_w} = \mathbf{\Phi} \cdot \mathbf{w}$ in this subspace has coordinates $[\mathbf{v_w}(s_1), \mathbf{v_w}(s_2), \ldots, \mathbf{v_w}(s_n)]$
- Vector $\mathbf{v_w}$ is fully specified by $\mathbf{w}$ (so we often say $\mathbf{w}$ to mean $\mathbf{v_w}$)

# Some more notation

- Denote $r(s, a)$ as the Expected Reward upon action $a$ in state $s$
- Denote $p(s, s', a)$ as the probability of transition $s \rightarrow s'$ upon action $a$
- Define

$$\mathbf{R}_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \cdot r(s, a)$$

$$\mathbf{P}_\pi(s, s') = \sum_{a \in \mathcal{A}} \pi(a|s) \cdot p(s, s', a)$$

- Denote $\mathbf{R}_\pi$ as the vector $[\mathbf{R}_\pi(s_1), \mathbf{R}_\pi(s_2), \ldots, \mathbf{R}_\pi(s_n)]$
- Denote $\mathbf{P}_\pi$ as the matrix $[\mathbf{P}_\pi(s_i, s_{i'})], 1 \leq i, i' \leq n$
- Denote $\gamma$ as the MDP discount factor

# Bellman operator $\mathbf{B}_\pi$

- Bellman operator $\mathbf{B}_\pi$ for policy $\pi$ operating on VF vector $\mathbf{v}$ defined as:

$$\mathbf{B}_\pi \mathbf{v} = \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{v}$$

- Note that $\mathbf{v}_\pi$ is the fixed point of operator $\mathbf{B}_\pi$ (meaning $\mathbf{B}_\pi \mathbf{v}_\pi = \mathbf{v}_\pi$)
- If we start with an arbitrary VF vector $\mathbf{v}$ and repeatedly apply $\mathbf{B}_\pi$, by Contraction Mapping Theorem, we will reach the fixed point $\mathbf{v}_\pi$
- This is the Dynamic Programming Policy Evaluation algorithm
- Monte Carlo without func approx also converges to $\mathbf{v}_\pi$ (albeit slowly)

# Projection operator $\mathbf{\Pi_\Phi}$

- First we define "distance" $d(\mathbf{v_1}, \mathbf{v_2})$ between VF vectors $\mathbf{v_1}, \mathbf{v_2}$
- Weighted by $\mu_\pi$ across the $n$ dimensions of $\mathbf{v_1}, \mathbf{v_2}$

$$d(\mathbf{v_1}, \mathbf{v_2}) = \sum_{i=1}^{n} \mu_\pi(s_i) \cdot (\mathbf{v_1}(s_i) - \mathbf{v_2}(s_i))^2 = (\mathbf{v_1} - \mathbf{v_2})^T \cdot \mathbf{D} \cdot (\mathbf{v_1} - \mathbf{v_2})$$

  where $\mathbf{D}$ is the square diagonal matrix consisting of $\mu_\pi(s_i), 1 \leq i \leq n$
- Projection operator for subspace spanned by $\mathbf{\Phi}$ is denoted as $\mathbf{\Pi_\Phi}$
- $\mathbf{\Pi_\Phi}$ performs an orthogonal projection of VF vector $\mathbf{v}$ on subspace $\mathbf{\Phi}$
- So, $\mathbf{\Pi_\Phi v}$ is the VF in subspace $\mathbf{\Phi}$ defined by $\arg\min_{\mathbf{w}} d(\mathbf{v}, \mathbf{v_w})$
- This is a weighted least squares regression with solution:

$$\mathbf{w} = (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{v}$$

- So, the Projection operator $\mathbf{\Pi_\Phi}$ can be written as:

$$\mathbf{\Pi_\Phi} = \mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D}$$

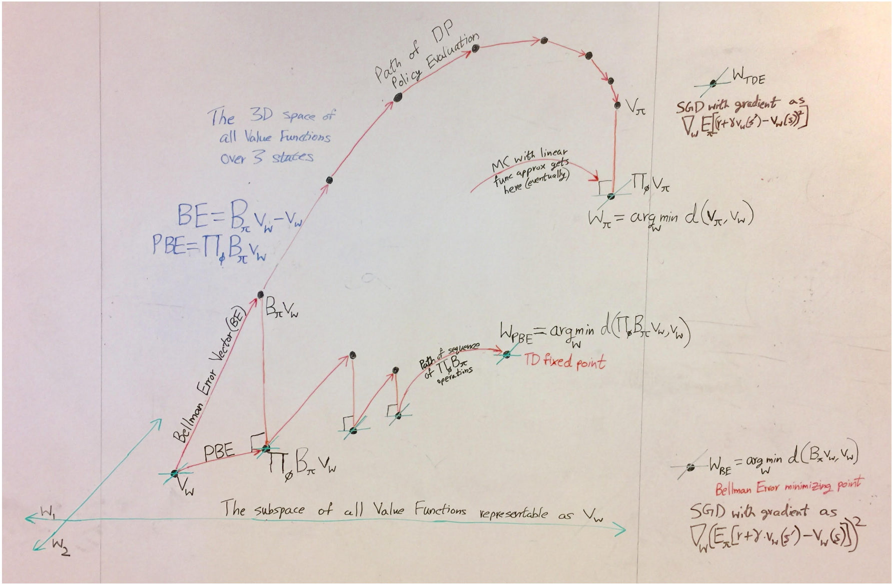
# 4 VF vectors of interest in the $\mathbf{\Phi}$ subspace

Note: We will refer to the $\mathbf{\Phi}$-subspace VF vectors by their weights $\mathbf{w}$

1. Projection $\mathbf{\Pi_\Phi v}_\pi$: $\mathbf{w}_\pi = \arg\min_{\mathbf{w}} d(\mathbf{v}_\pi, \mathbf{v}_{\mathbf{w}})$
   - This is the VF we seek when doing linear function approximation
   - Because it is the VF vector "closest" to $\mathbf{v}_\pi$ in the $\mathbf{\Phi}$ subspace
   - Monte-Carlo with linear func approx will (slowly) converge to $\mathbf{w}_\pi$
2. Bellman Error (BE)-minimizing: $\mathbf{w}_{BE} = \arg\min_{\mathbf{w}} d(\mathbf{B}_\pi \mathbf{v}_{\mathbf{w}}, \mathbf{v}_{\mathbf{w}})$
   - This is the solution to a linear system (covered later)
   - In model-free setting, Residual Gradient TD Algorithm (covered later)
   - Cannot learn if we can only access features, and not underlying states

# 4 VF vectors of interest in the $\mathbf{\Phi}$ subspace (continued)

3. Projected Bellman Error (PBE)-minimizing:
   $\mathbf{w}_{PBE} = \arg\min_{\mathbf{w}} d((\mathbf{\Pi_\Phi} \cdot \mathbf{B}_\pi)\mathbf{v_w}, \mathbf{v_w})$
   - The minimum is 0, i.e., $\mathbf{\Phi} \cdot \mathbf{w}_{PBE}$ is the fixed point of operator $\mathbf{\Pi_\Phi} \cdot \mathbf{B}_\pi$
   - This fixed point is the solution to a linear system (covered later)
   - Alternatively, if we start with an arbitrary $\mathbf{v_w}$ and repeatedly apply $\mathbf{\Pi_\Phi} \cdot \mathbf{B}_\pi$, we will converge to $\mathbf{\Phi} \cdot \mathbf{w}_{PBE}$
   - This is a DP-like process with approximation - repeatedly thrown out of the $\mathbf{\Phi}$ subspace (applying Bellman operator $\mathbf{B}_\pi$), followed by landing back in the $\mathbf{\Phi}$ subspace (applying Projection operator $\mathbf{\Pi_\Phi}$)
   - In model-free setting, Gradient TD Algorithms (covered later)
4. Temporal Difference Error (TDE)-minimizing:
   $\mathbf{w}_{TDE} = \arg\min_{w} \mathbb{E}_\pi[\delta^2]$
   - $\delta$ is the TD error
   - Minimizes the expected square of the TD error when following policy $\pi$
   - Naive Residual Gradient TD Algorithm (covered later)


Path of DP Policy Evaluation
The 3D space of all Value Functions over 3 states
$V_\pi$
$W_{TDE}$
SGD with gradient as $\nabla_w E_\pi[(r+\gamma v_w(s')-v_w(s))^2]$
MC with linear func approx gets here (eventually)
$\Pi_\phi V_\pi$
$W_\pi = \arg\min_w d(V_\pi, V_w)$
$BE = B_\pi V_w - V_w$
$PBE = \Pi_\phi B_\pi V_w$
$B_\pi V_w$
Bellman Error Vector (BE)
$W_{PBE} = \arg\min_w d(\Pi_\phi B_\pi V_w, V_w)$
TD fixed point
Path of sequence of $\Pi_\phi B_\pi$ operations
PBE
$\Pi_\phi B_\pi V_w$
$V_w$
$W_{BE} = \arg\min_w d(B_\pi V_w, V_w)$
Bellman Error minimizing point
SGD with gradient as $\nabla_w (E_\pi[r+\gamma \cdot v_w(s')-V_w(s)])^2$
$W_1$
$W_2$
The subspace of all Value Functions representable as $V_w$

# Solution of $\mathbf{w}_{BE}$ with a Linear System Formulation

$$
\begin{aligned}
\mathbf{w}_{BE} &= \arg\min_{\mathbf{w}} d(\mathbf{v}_{\mathbf{w}}, \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{v}_{\mathbf{w}}) \\
&= \arg\min_{\mathbf{w}} d(\mathbf{\Phi} \cdot \mathbf{w}, \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}) \\
&= \arg\min_{\mathbf{w}} d(\mathbf{\Phi} \cdot \mathbf{w} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}, \mathbf{R}_\pi) \\
&= \arg\min_{\mathbf{w}} d((\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}) \cdot \mathbf{w}, \mathbf{R}_\pi)
\end{aligned}
$$

This is a weighted least-squares linear regression of $\mathbf{R}_\pi$ versus $\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}$ with weights $\mu_\pi$, whose solution is:

$$
w_{BE} = ((\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi})^T \cdot \mathbf{D} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}))^{-1} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi})^T \cdot \mathbf{D} \cdot \mathbf{R}_\pi
$$

# Solution of $\mathbf{w}_{PBE}$ with a Linear System Formulation

$\mathbf{\Phi} \cdot \mathbf{w}_{PBE}$ is the fixed point of operator $\mathbf{\Pi_\Phi} \cdot \mathbf{B}_\pi$. We know:

$$\mathbf{\Pi_\Phi} = \mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D}$$

$$\mathbf{B}_\pi \mathbf{v} = \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{v}$$

Therefore,

$$\mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}_{PBE}) = \mathbf{\Phi} \cdot \mathbf{w}_{PBE}$$

Since columns of $\mathbf{\Phi}$ are assumed to be independent (full rank),

$$\begin{aligned}
(\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}_{PBE}) &= \mathbf{w}_{PBE} \\
\mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}_{PBE}) &= \mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi} \cdot \mathbf{w}_{PBE} \\
\mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}) \cdot \mathbf{w}_{PBE} &= \mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{R}_\pi
\end{aligned}$$

This is a square linear system of the form $\mathbf{A} \cdot \mathbf{w}_{PBE} = \mathbf{b}$ whose solution is:

$$\mathbf{w}_{PBE} = \mathbf{A}^{-1} \cdot \mathbf{b} = (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}))^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{R}_\pi$$

# Model-Free Learning of $\mathbf{w}_{PBE}$

- How do we construct matrix $\mathbf{A} = \mathbf{\Phi}^T \cdot \mathbf{D} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi})$ and vector $\mathbf{b} = \mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{R}_\pi$ without a model?
- Following policy $\pi$, each time we perform a model-free transition from $s$ to $s'$ getting reward $r$, we get a sample estimate of $\mathbf{A}$ and $\mathbf{b}$
- Estimate of $\mathbf{A}$ is the outer-product of vectors $\phi(s)$ and $\phi(s) - \gamma \cdot \phi(s')$
- Estimate of $\mathbf{b}$ is scalar $r$ times vector $\phi(s)$
- Average these estimates across many such model-free transitions
- This algorithm is called Least Squares Temporal Difference (LSTD)
- Alternative: Semi-Gradient Temporal Difference (TD) descent
- This semi-gradient descent converges to $\mathbf{w}_{PBE}$ with weight updates:

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

- $r + \gamma \cdot w^T \cdot \phi(s') - w^T \cdot \phi(s)$ is the TD error, denoted $\delta$

# Naive Residual Gradient Algorithm to solve for $\mathbf{w}_{TDE}$

- We defined $\mathbf{w}_{TDE}$ as the vector in the $\mathbf{\Phi}$ subspace that minimizes the expected square of the TD error $\delta$ when following policy $\pi$.

$$\mathbf{w}_{TDE} = \arg\min_{\mathbf{w}} \sum_{s \in \mathcal{S}} \mu_\pi(s) \sum_{r,s'} prob_\pi(r, s'|s) \cdot (r + \gamma \cdot \mathbf{w}^T \cdot \phi(s') - \mathbf{w}^T \cdot \phi(s))^2$$

- To perform SGD, we have to estimate the gradient of the expected square of TD error by sampling
- The weight update for each sample in the SGD will be:

$$\begin{aligned} \Delta w &= -\frac{1}{2}\alpha \cdot \nabla_w (r + \gamma \cdot w^T \cdot \phi(s') - w^T \cdot \phi(s))^2 \\ &= \alpha \cdot (r + \gamma \cdot w^T \cdot \phi(s') - w^T \cdot \phi(s)) \cdot (\phi(s) - \gamma \cdot \phi(s')) \end{aligned}$$

- This algorithm (named *Naive Residual Gradient*) converges robustly, but not to a desirable place

# Residual Gradient Algorithm to solve for $\mathbf{w}_{BE}$

- We defined $\mathbf{w}_{BE}$ as the vector in the $\mathbf{\Phi}$ subspace that minimizes BE
- But BE for a state is the expected TD error in that state
- So we want to do SGD with gradient of square of expected TD error

$$\begin{aligned}
\Delta w &= -\frac{1}{2}\alpha \cdot \nabla_w (\mathbb{E}_\pi[\delta])^2 \\
&= -\alpha \cdot \mathbb{E}_\pi[r + \gamma \cdot w^T \cdot \phi(s') - w^T \cdot \phi(s)] \cdot \nabla_w \mathbb{E}_\pi[\delta] \\
&= \alpha \cdot (\mathbb{E}_\pi[r + \gamma \cdot w^T \cdot \phi(s')] - w^T \cdot \phi(s)) \cdot (\phi(s) - \gamma \cdot \mathbb{E}_\pi[\phi(s')])
\end{aligned}$$

- This is called the *Residual Gradient* algorithm
- Requires two independent samples of $s'$ transitioning from $s$
- In that case, converges to $\mathbf{w}_{BE}$ robustly (even for non-linear approx)
- But it is slow, and doesn't converge to a desirable place
- Cannot learn if we can only access features, and not underlying states

# Gradient TD Algorithms to solve for $\mathbf{w}_{PBE}$

- For on-policy linear func approx, semi-gradient TD works
- For non-linear func approx or off-policy, we need Gradient TD
  - GTD: The original Gradient TD algorithm
  - GTD-2: Second-generation GTD
  - TDC: TD with Gradient correction
- We need to set up the loss function whose gradient will drive SGD
- $\mathbf{w}_{PBE} = \arg\min_{\mathbf{w}} d(\mathbf{\Pi_\Phi B}_\pi \mathbf{v_w}, \mathbf{v_w}) = \arg\min_{\mathbf{w}} d(\mathbf{\Pi_\Phi B}_\pi \mathbf{v_w}, \mathbf{\Pi_\Phi v_w})$
- So we define the loss function (denoting $\mathbf{B}_\pi \mathbf{v_w} - \mathbf{v_w}$ as $\delta_{\mathbf{w}}$) as:

$$\mathcal{L}(\mathbf{w}) = (\mathbf{\Pi_\Phi} \delta_{\mathbf{w}})^T \cdot \mathbf{D} \cdot (\mathbf{\Pi_\Phi} \delta_{\mathbf{w}}) = \delta_{\mathbf{w}}{}^T \cdot \mathbf{\Pi}_{\mathbf{\Phi}}^T \cdot \mathbf{D} \cdot \mathbf{\Pi_\Phi} \cdot \delta_{\mathbf{w}}$$

$$= \delta_{\mathbf{w}}{}^T \cdot (\mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D})^T \cdot \mathbf{D} \cdot (\mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D}) \cdot \delta_{\mathbf{w}}$$

$$= \delta_{\mathbf{w}}{}^T \cdot (\mathbf{D} \cdot \mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T) \cdot \mathbf{D} \cdot (\mathbf{\Phi} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{D}) \cdot \delta_{\mathbf{w}}$$

$$= (\delta_{\mathbf{w}}{}^T \cdot \mathbf{D} \cdot \mathbf{\Phi}) \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi}) \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})$$

$$= (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^T \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})$$

# TDC Algorithm to solve for $\mathbf{w}_{PBE}$

We derive the TDC Algorithm based on $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = 2 \cdot (\nabla_{\mathbf{w}}(\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^T) \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})$$

Now we express each of these 3 terms as expectations of model-free transitions $s \xrightarrow{\mu} (r, s')$, denoting $r + \gamma \cdot \mathbf{w}^T \cdot \phi(s') - \mathbf{w}^T \cdot \phi(s)$ as $\delta$

- $\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}} = \mathbb{E}[\delta \cdot \phi(s)]$
- $\nabla_{\mathbf{w}}(\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^T = \nabla_{\mathbf{w}}(\mathbb{E}[\delta \cdot \phi(s)])^T = \mathbb{E}[(\nabla_{\mathbf{w}}\delta) \cdot \phi(s)^T] = \mathbb{E}[(\gamma \cdot \phi(s') - \phi(s)) \cdot \phi(s)^T]$
- $\mathbf{\Phi}^T \cdot \mathbf{D} \cdot \mathbf{\Phi} = \mathbb{E}[\phi(s) \cdot \phi(s)^T]$

Substituting, we get:

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = 2 \cdot \mathbb{E}[(\gamma \cdot \phi(s') - \phi(s)) \cdot \phi(s)^T] \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]$$

# Weight Updates of TDC Algorithm

$$\Delta w = -\frac{1}{2}\alpha \cdot \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$
$$= \alpha \cdot \mathbb{E}[(\phi(s) - \gamma \cdot \phi(s')) \cdot \phi(s)^T] \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]$$
$$= \alpha \cdot (\mathbb{E}[\phi(s) \cdot \phi(s)^T] - \gamma \cdot \mathbb{E}[\phi(s') \cdot \phi(s)^T]) \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]$$
$$= \alpha \cdot (\mathbb{E}[\delta \cdot \phi(s)] - \gamma \cdot \mathbb{E}[\phi(s') \cdot \phi(s)^T] \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)])$$
$$= \alpha \cdot (\mathbb{E}[\delta \cdot \phi(s)] - \gamma \cdot \mathbb{E}[\phi(s') \cdot \phi(s)^T] \cdot \theta)$$

where $\theta = \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]$ is the solution to a weighted least-squares linear regression of $\mathbf{B}_\pi \mathbf{v} - \mathbf{v}$ against $\mathbf{\Phi}$, with weights as $\mu_\pi$.

**Cascade Learning: Update both** $w$ **and** $\theta$ ($\theta$ converging faster)

- $\Delta w = \alpha \cdot \delta \cdot \phi(s) - \alpha \cdot \gamma \cdot \phi(s') \cdot (\theta^T \cdot \phi(s))$
- $\Delta\theta = \beta \cdot (\delta - \theta^T \cdot \phi(s)) \cdot \phi(s)$

Note: $\theta^T \cdot \phi(s)$ operates as estimate of TD error $\delta$ for current state $s$