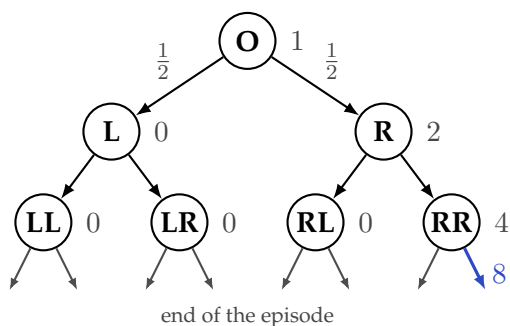


Lecture 13: Function Approximation I: Features and Linear Values

How to use these notes. Board content in board order: figures, equations, tables, one-line statements. Today's task is *policy evaluation when the table is too big*: the learner keeps d numbers instead of one per state, and the episodes have to fit them. The note stands on its own: every symbol is defined where it is first used. Reading: [Sutton and Barto \(2018, Sections 9.1–9.4\)](#); optional, [Tsitsiklis and Van Roy \(1997\)](#) and [Szepesvári \(2010, Section 3.2.1\)](#). Throughout the course, the notes draw on the public materials of Stanford CS 234 ([Brunskill, 2026](#)), UIUC CS 443 ([Jiang, 2026](#)), and Cornell CS 4789 ([Sun, 2025](#)); some examples are adapted from them.

The Task: Policy Evaluation When the Table Is Too Big

A prize tree: three steps, a fair coin at each (the policy π), the prize 8 behind the rightmost last step, $\gamma = 1$. Wanted: V^π at seven states, with four numbers.



The prize tree under π . Gray: hidden from the learner, the truth V^π beside each state, (1; 0, 2; 0, 0, 0, 4) in the order O; L, R; LL, LR, RL, RR.

Task	States $ \mathcal{S} $
the tree, 3 steps	7
the tree, H steps	$2^H - 1$
backgammon	10^{20}
Go, 19×19	10^{170}
one 84×84 screen, 256 grays	256^{7056}

One entry per state, one entry moved a step. From the third row on ([Tesauro, 1995](#); [Tromp and Farnebäck, 2007](#)), no state is seen twice.

Policy evaluation: what is wanted (stated) (Sutton and Barto, 2018, Section 3.5). A Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$: action a in state s pays a reward with mean $r(s, a)$ and moves to s' w.p. $P(s' | s, a)$; $\gamma \in (0, 1]$, with $\gamma = 1$ allowed when every episode ends. Fix π ; $r^\pi(s)$, $P^\pi(s' | s)$: the pay and the move of the step leaving s .

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots, \quad V^\pi(s) := \mathbb{E}[G_t | S_t = s] \text{ under } \pi$$

the return to the end of the episode, and its mean: the value of π at s

$$(\mathcal{T}^\pi V)(s) := r^\pi(s) + \gamma \sum_{s'} P^\pi(s' | s) V(s'); \quad V^\pi = \mathcal{T}^\pi V^\pi, \quad V(\text{end}) = 0$$

the backup of any table V ; Bellman's equation: V^π is its only fixed point

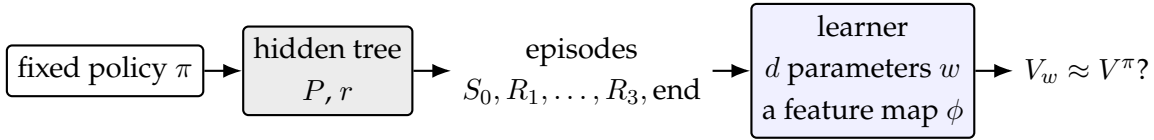
$$V(\text{RR}) = \frac{1}{2} \cdot 8 + \frac{1}{2} \cdot 0 = 4, \quad V(\text{LL}) = V(\text{LR}) = V(\text{RL}) = 0$$

on the tree, from the bottom: the last step pays the prize or nothing

$$V(\text{R}) = \frac{1}{2} V(\text{RL}) + \frac{1}{2} V(\text{RR}) = 2, \quad V(\text{L}) = 0, \quad V(\text{O}) = \frac{1}{2} V(\text{L}) + \frac{1}{2} V(\text{R}) = 1$$

$V^\pi = (1; 0, 2; 0, 0, 0, 4)$: backward induction, with P and r in hand

Today P and r are hidden, and the learner has no room for seven numbers.



Policy evaluation with features: π acts, the tree answers, the learner moves d numbers that stand for every state; it never sees P or r .

The data, the template, and the new constraint.

$S_0 = \text{O}, R_1, S_1, R_2, S_2, R_3, \text{end}; \quad S_{t+1} \sim P(\cdot | S_t, \pi(S_t)), \quad R_{t+1}$ the pay of the step
one episode of π on the tree: three steps, then a fresh start at O; no action written ($A_t = \pi(S_t)$, the coin)

$$V(s) \leftarrow V(s) + \alpha (\text{target} - V(s))$$

the template with a table (stated): one entry moves a fraction α toward the target

$$\text{target} = G_t \text{ (Monte Carlo)} \quad \text{or} \quad \text{target} = R_{t+1} + \gamma V(S_{t+1}) \text{ (TD(0))}, \quad V(\text{end}) = 0$$

the two targets (stated): the return, or one reward and the table's guess at the next state

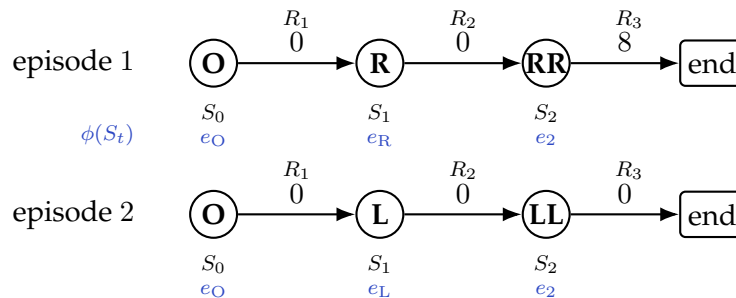
$\gamma = 1$ on the tree: every episode ends after three steps, and the return is a plain sum

every formula below keeps γ , as on a chain that never ends; the numbers set $\gamma = 1$

today: no table; $w \in \mathbb{R}^d$ and a feature vector $\phi(s) \in \mathbb{R}^d$ per state, $d = 4$ on the tree

the learner writes $V_w(s) = w^\top \phi(s)$ and may move w only

Known to the learner: the episodes, $\gamma, \phi(s)$. Hidden: P, r, V^π .



Two episodes, used by hand below: the prize, then a left turn and nothing. Under each state, its feature vector in the layer class: RR and LL share e_2 .

When the table is too big to fill, how do we evaluate π with d numbers: what do we fit, how do the two targets move the d numbers, where does each stop, and why is the stop of TD not the best fit?

Today's targets.

1. Features: $V_w(s) = w^\top \phi(s)$, $\nabla_w V_w = \phi$; the table is $\phi(s) = e_s$; a group shares one number: four numbers on the tree, two ways.
2. Best: $\overline{VE}(w) = \sum_s \mu(s)(V^\pi(s) - V_w(s))^2$, μ the visits per episode; the layer class at $(1; 0, 2; 1, 1, 1, 1)$, $\overline{VE} = 3$; the dead-end class exact.
3. One template, two targets: $w \leftarrow w + \alpha(\text{target} - V_w(S_t))\nabla_w V_w(S_t)$, target G_t or $R_{t+1} + \gamma V_w(S_{t+1})$; two episodes by hand.
4. Where each stops: the same d equations, a different bracket; $w_{\text{TD}} = (1, 1, 1, 1)$ against $w_{\text{LS}} = (1, 0, 2, 1)$, $\overline{VE}$ 4 against 3.
5. By computer from 4: at episode 20 the dead-end class 5.28, the table 19.67; at 2,000 the floors 4.53, 3.89, 0.49.

Features and Linear Values

Linear values (Sutton and Barto, 2018, Sections 9.3–9.4). A *feature* is a number computed from the state; d of them make the feature vector $\phi(s) = (\phi_1(s), \dots, \phi_d(s)) \in \mathbb{R}^d$.

$$V_w(s) := w^\top \phi(s) = \sum_{i=1}^d w_i \phi_i(s), \quad w \in \mathbb{R}^d \quad \text{the fit: } d \text{ parameters, the same } w \text{ in every state}$$

$$\nabla_w V_w(s) = \phi(s) \quad \text{how much each parameter moves } V_w(s): \text{ the feature itself}$$

$$\mathcal{F} := \{V_w : w \in \mathbb{R}^d\}$$

the class: the value tables the learner can write down, a d -dimensional plane in $\mathbb{R}^{|\mathcal{S}|}$

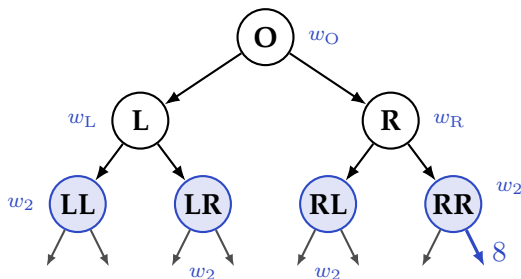
$$\phi(s) = e_s \quad (d = |\mathcal{S}|) : \quad V_w(s) = w_s$$

one-hot features: w is the table; every formula below reduces to the tabular one

$$\phi(s) = e_{g(s)} \quad (\text{groups } 1, \dots, d) : \quad V_w(s) = w_{g(s)}, \quad \nabla_w V_w(s) = e_{g(s)}$$

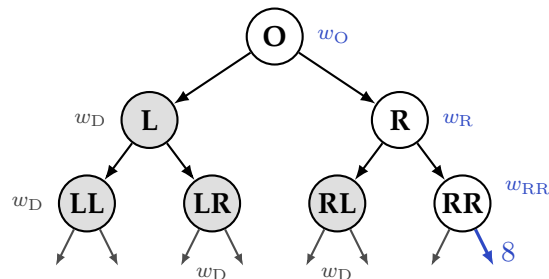
state aggregation: one number per group of states; the table is one state per group

Generalization: moving w moves every s' whose $\phi(s')$ overlaps $\phi(s)$, a whole group at once; the price: $\mathcal{F}$ need not hold V^π . The updates below need two things of V_w , its value and $\nabla_w V_w(s)$: any differentiable V_w , a network included, supplies both; linear is the case done by hand and the case the theorem covers.



The layer class: $w = (w_O, w_L, w_R, w_2)$, one number for the bottom layer (blue);

$$V_w = (w_O; w_L, w_R; w_2, w_2, w_2, w_2).$$



The dead-end class: $w = (w_O, w_R, w_{RR}, w_D)$, one number for the four dead ends (gray), from which the prize is out of reach;

$$V_w = (w_O; w_D, w_R; w_D, w_D, w_D, w_{RR}).$$

Example: is the truth $V^\pi = (1; 0, 2; 0, 0, 0, 4)$ in the class?

layer: $(w_O; w_L, w_R; w_2, w_2, w_2, w_2) = (1; 0, 2; 0, 0, 0, 4)$? no

w_2 would have to be 0 and 4: the bottom layer must share one number

dead-end: $(w_O; w_D, w_R; w_D, w_D, w_D, w_{RR}) = (1; 0, 2; 0, 0, 0, 4)$ at $w = (1, 2, 4, 0)$ yes

four numbers say what seven did; which w_2 is best for the layer class: next

- Depth H : $H + 1$ numbers against $2^H - 1$ entries, a table that never grows; the features,

the depth and whether a left step was taken, are read off the path.

- On a real task the features are the design: the columns of Tetris, the points of backgammon, the pixels of a screen. The same formulas apply.

What “Best” Means: the Episodes’ Weighting

The weighted error (Sutton and Barto, 2018, Section 9.2).

$\mu(s) :=$ the expected number of visits to s per episode

the weighting of the episodes (stated); on the tree, the probability of passing through s

$$\mu = (1; \frac{1}{2}, \frac{1}{2}; \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$$

O starts every episode; one coin decides L or R; two coins decide the bottom state; each layer sums to 1

$$\overline{\text{VE}}(w) := \sum_s \mu(s) (V^\pi(s) - V_w(s))^2$$

the error of a fit, each state counted as often as the episodes visit it

$$w_{\text{LS}} := \underset{w}{\operatorname{argmin}} \overline{\text{VE}}(w), \quad \Pi V^\pi := V_{w_{\text{LS}}}$$

the best fit in $\mathcal{F}$: the projection of the truth onto the plane, under μ

The weighting belongs to the episodes, not to the learner; scaling μ moves nothing below.

Example: the best fit in the layer class.

$$\overline{\text{VE}}(w) = (1 - w_{\text{O}})^2 + \frac{1}{2}(0 - w_{\text{L}})^2 + \frac{1}{2}(2 - w_{\text{R}})^2 + \frac{1}{4}[3(0 - w_2)^2 + (4 - w_2)^2]$$

seven terms, three of them equal

$$\partial_{w_{\text{O}}} = 0 : w_{\text{O}} = 1; \quad \partial_{w_{\text{L}}} = 0 : w_{\text{L}} = 0; \quad \partial_{w_{\text{R}}} = 0 : w_{\text{R}} = 2$$

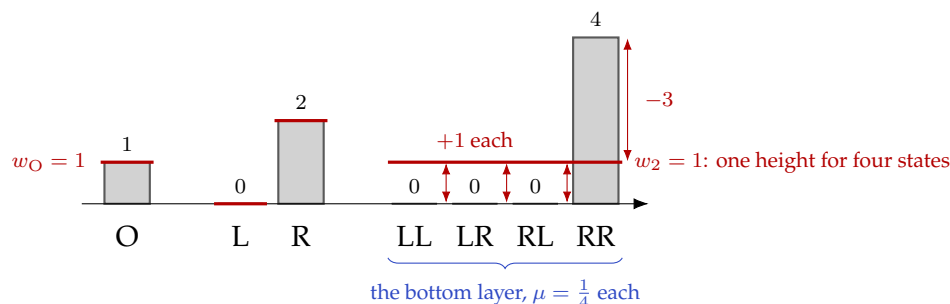
a state alone in its group gets its truth

$$\partial_{w_2} = 0 : 3(0 - w_2) + (4 - w_2) = 0 \Rightarrow w_2 = \frac{0+0+0+4}{4} = 1$$

the mean of the four truths, each weighed $\frac{1}{4}$

$$V_{w_{\text{LS}}} = (1; 0, 2; 1, 1, 1, 1), \quad \overline{\text{VE}} = \frac{1}{4}(1 + 1 + 1 + 9) = 3$$

errors (0; 0, 0; +1, +1, +1, -3); the dead-end class: $\overline{\text{VE}} = 0$



The truth (gray) against the layer class (red): the best shared height is the mean, 1; three errors of 1 against one of 3.

Example: why the weights. The coin falls right w.p. $\frac{2}{3}$ and the prize is 27.

$$V^\pi = (8; 0, 12; 0, 0, 0, 18), \quad \mu(\text{LL}), \mu(\text{LR}), \mu(\text{RL}), \mu(\text{RR}) = \frac{1}{9}, \frac{2}{9}, \frac{2}{9}, \frac{4}{9}$$

$$18 = \frac{2}{3} \cdot 27, 12 = \frac{2}{3} \cdot 18, 8 = \frac{2}{3} \cdot 12; \text{two coins decide the bottom state}$$

$$w_2 = \frac{1}{9} \cdot 0 + \frac{2}{9} \cdot 0 + \frac{2}{9} \cdot 0 + \frac{4}{9} \cdot 18 = 8 \neq \frac{0+0+0+18}{4} = 4.5$$

RR is visited four times as often as LL, so its error counts four times

- One-hot: $\overline{VE} = 0$ at the table V^π , no best-fit question. A smaller class: the weighting is part of the answer.
- $\overline{VE}$ is what the learner would minimize and cannot compute: V^π is hidden. The two targets stand in for it, below.

Monte Carlo as Least Squares

Gradient Monte Carlo (Sutton and Barto, 2018, Section 9.3).

$$\ell(v; y) = \frac{1}{2}(v - y)^2, \quad v \leftarrow v - \alpha \partial_v \ell(v; y) = v + \alpha(y - v)$$

fixed target y ; $v = V(S_t)$ gives the table update

$$\ell_t(w) = \frac{1}{2}(V_w(S_t) - G_t)^2$$

Monte Carlo: target G_t , independent of w

$$w \leftarrow w - \alpha \nabla_w \ell_t(w) = w + \alpha (G_t - V_w(S_t)) \nabla_w V_w(S_t) \quad \text{chain rule: any differentiable } V_w$$

features: $w \leftarrow w + \alpha (G_t - V_w(S_t)) \phi(S_t)$

$\nabla_w V_w = \phi$: the error, spread over the features of S_t ; a group: $w_g \leftarrow w_g + \alpha(G_t - w_g)$

$$\mathbb{E}[(G_t - V_w(S_t)) \nabla_w V_w(S_t)] = \sum_s \mu(s) (V^\pi(s) - V_w(s)) \nabla_w V_w(s) = -\frac{1}{2} \nabla \overline{VE}(w)$$

$\mathbb{E}[G_t | S_t = s] = V^\pi(s)$ and S_t weighted by μ : a step down $\overline{VE}$, for any V_w

$$w_t \rightarrow w_{LS} \quad \text{under } \sum \alpha = \infty, \sum \alpha^2 < \infty \text{ (stated)}$$

stochastic gradient descent on $\overline{VE}$ (Robbins and Monro, 1951): least squares on the returns

One-hot: the running average at $\alpha = 1/n$; a group: the running average of every return from any state of the group. A nonlinear V_w reaches a stationary point of $\overline{VE}$, not always its minimum (stated).

Example: gradient Monte Carlo by hand. $\alpha = \frac{1}{2}$, $w = 0$, the layer class, the two episodes of the figure.

episode 1 : $G = 8$ at O, R, RR

$\gamma = 1$: one return for the whole episode, the prize

$$w_O \leftarrow 0 + \frac{1}{2}(8 - 0) = 4, \quad w_R \leftarrow 0 + \frac{1}{2}(8 - 0) = 4, \quad w_2 \leftarrow 0 + \frac{1}{2}(8 - 0) = 4$$

the bottom layer reads 4 at all four states

episode 2 : $G = 0$ at O, L, LL

$$w_O \leftarrow 4 + \frac{1}{2}(0 - 4) = 2, \quad w_L \leftarrow 0 + \frac{1}{2}(0 - 0) = 0, \quad w_2 \leftarrow 4 + \frac{1}{2}(0 - 4) = 2$$

L stays at its truth: its target is its own return

$$V_w = (2; 0, 4; 2, 2, 2, 2), \quad \text{the table, same episodes: } (2; 0, 4; 0, 0, 0, 4)$$

they differ on the bottom layer only

- Unbiased target, a whole episode of noise, three steps late: as with a table. New: the destination is the best fit, not the truth.
- The zero of the mean step, $\sum_s \mu(s) \phi(s) (V^\pi(s) - V_w(s)) = 0$: d equations, the normal equations of least squares; on the tree, the best-fit Example.

Semi-Gradient TD(0): the Template With Features

Semi-gradient TD(0) (Sutton, 1988), (Sutton and Barto, 2018, Section 9.3).

- Input: the episodes of π on the tree; the features ϕ ; a step size α . Output: w , meant to make $V_w \approx V^\pi$.
- Start: $w_0 = 0$. w_t is the parameter vector before the update of step t .
- Each step t : observe (S_t, R_{t+1}, S_{t+1}) , $V_w(\text{end}) = 0$:

$$\begin{aligned}
 y_t &= R_{t+1} + \gamma V_{w_t}(S_{t+1}), & \delta_t &= y_t - V_{w_t}(S_t) && \text{target and TD error} \\
 \ell_t(w) &= \frac{1}{2} (V_w(S_t) - y_t)^2 && \text{hold } y_t \text{ fixed when differentiating} \\
 w_{t+1} &= w_t - \alpha \nabla_w \ell_t(w_t) = w_t + \alpha \delta_t \nabla_w V_w(S_t) \big|_{w=w_t} && \text{chain rule} \\
 w_{t+1} &= w_t + \alpha \delta_t \phi(S_t), & \delta_t &= R_{t+1} + \gamma w_t^\top \phi(S_{t+1}) - w_t^\top \phi(S_t) \\
 &&& \text{with features: the form used by hand below; a group: } w_g \leftarrow w_g + \alpha \delta_t
 \end{aligned}$$

- Memory d ; work one value and one gradient of V_w a step, $O(d)$ with features. One-hot: TD(0) on the table.

Semi: differentiate the prediction while holding y_t fixed. The target changes with w_t ; this is not gradient descent on $\overline{VE}$ or on the full squared TD error.

Example: semi-gradient TD(0) by hand. $\alpha = \frac{1}{2}$, $\gamma = 1$, $w = 0$, the layer class, the two episodes of the figure; $\delta_t = R_{t+1} + V_w(S_{t+1}) - V_w(S_t)$.

$$t = 0, (O, 0, R) : \quad \delta_0 = 0 + w_R - w_O = 0 \quad \text{nothing moves: the target reads 0, the fit reads 0}$$

$$t = 1, (R, 0, RR) : \quad \delta_1 = 0 + w_2 - w_R = 0$$

$$t = 2, (RR, 8, \text{end}) : \quad \delta_2 = 8 + 0 - w_2 = 8, \quad w_2 \leftarrow 0 + \frac{1}{2} \cdot 8 = 4$$

one visit to RR, and LL, LR, RL read 4 too

$$\text{after episode 1 : } V_w = (0; 0, 0; 4, 4, 4, 4), \quad \text{the table: } (0; 0, 0; 0, 0, 0, 4)$$

$$t = 3, (O, 0, L) : \quad \delta_3 = 0 + w_L - w_O = 0$$

$$t = 4, (L, 0, LL) : \quad \delta_4 = 0 + w_2 - w_L = 4 - 0 = 4, \quad w_L \leftarrow 0 + \frac{1}{2} \cdot 4 = 2$$

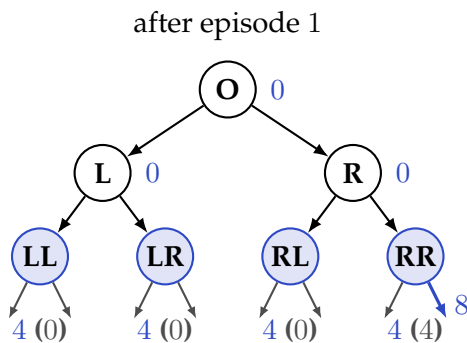
the target at L reads the shared 4; the table's target read $0 + 0 = 0$, L stays 0

$$t = 5, (LL, 0, \text{end}) : \quad \delta_5 = 0 + 0 - w_2 = -4, \quad w_2 \leftarrow 4 + \frac{1}{2}(-4) = 2$$

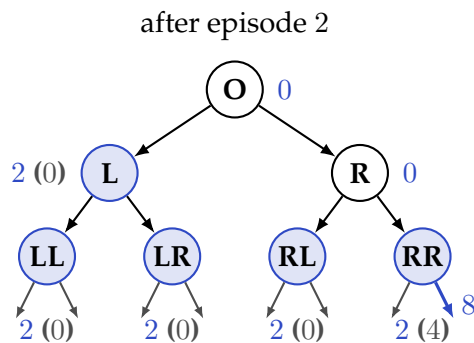
LL pulls the shared number down, RR with it

$$\text{after episode 2 : } V_w = (0; 2, 0; 2, 2, 2, 2), \quad \text{the table: } (0; 0, 0; 0, 0, 0, 4)$$

the truth: (1; 0, 2; 0, 0, 0, 4); a dead end reads 2



The fit in blue, the table in gray: one visit to RR, and three states that never saw the prize read 4.



Generalization cuts both ways: the dead end L reads 2, fed through its target by RR, a state it can never reach.

Example: the dead-end class, one step. Every parameter starts at 4, a guess; observe (LL, 0, end), $\alpha = \frac{1}{2}$.

$$\delta = 0 + 0 - w_D = -4, \quad w_D \leftarrow 4 + \frac{1}{2}(-4) = 2 : \quad V_w = (4; 2, 4; 2, 2, 2, 4)$$

L, LL, LR, RL read 2 at once: the tied states agree, generalization the right way

the table: (4; 4, 4, 4; 2, 4, 4, 4)

LL alone; L, LR, RL wait for their own visits

Check: where is w pulled next? With P , which the learner lacks, from $V_w = (0; 2, 0; 2, 2, 2, 2)$:

$$\mathcal{T}^\pi V_w = \left(\frac{1}{2}(w_L + w_R); w_2, w_2; 0, 0, 0, 4\right) = (1; 2, 2; 0, 0, 0, 4)$$

one backup of the fit: the mean of the children, or the last pay

mean step per unit of α , group by group: $\sum_{s \in g} \mu(s) ((\mathcal{T}^\pi V_w)(s) - V_w(s))$

$$w_O: 1 \cdot (1 - 0) = 1, \quad w_L: \frac{1}{2}(2 - 2) = 0$$

L is wrong at 2 and does not move: its target is the number it already matches

$$w_R: \frac{1}{2}(2 - 0) = 1, \quad w_2: \frac{1}{4}[3(0 - 2) + (4 - 2)] = -1$$

R is pulled up by the shared number; the shared number is pulled down by three dead ends

- $\alpha = \frac{1}{2}$ moves half the error here because $\|\phi(s)\|^2 = 1$; longer feature vectors need a smaller step ($\alpha \|\phi\|^2 \leq 1$, or an update overshoots its target).
- Two episodes moved three numbers; where they are pulled in the long run, and whether that is the best fit: the derivation.
- A neural network as V_w changes one factor in the box, $\nabla_w V_w(S_t)$, by automatic differentiation; the target stays frozen; the theorem below needs a linear V_w .

Where Each Method Stops: the TD Fixed Point

Derivation: from the update to its fixed point (Sutton and Barto, 2018, Section 9.4). A fixed point has zero mean update. Hold w fixed and average over visits; their state probabilities are proportional to $\mu(s)$.

$$\Delta w := w_{\text{new}} - w = \alpha (y_t - V_w(S_t)) \phi(S_t) \quad \text{the linear update already used above; } y_t \text{ is the target}$$

$$\mathbb{E}[\Delta w \mid S_t = s] = \alpha (\mathbb{E}[y_t \mid S_t = s] - V_w(s)) \phi(s)$$

fix the state: the prediction and features are fixed; average the target

$$\mathbb{E}[\Delta w] = \frac{\alpha}{\sum_s \mu(s)} \sum_s \mu(s) \phi(s) (\mathbb{E}[y_t \mid S_t = s] - V_w(s))$$

average over states: weight by visits and divide by their total

$$\text{MC: } \sum_s \mu(s) \phi(s) (V^\pi(s) - V_w(s)) = 0 \iff w = w_{\text{LS}}$$

set the mean update to zero; target G_t has mean $V^\pi(s)$: the best-fit equations

$$\text{TD: } \sum_s \mu(s) \phi(s) ((\mathcal{T}^\pi V_w)(s) - V_w(s)) = 0 \iff w = w_{\text{TD}}$$

target $R_{t+1} + \gamma V_w(S_{t+1})$ has mean $(\mathcal{T}^\pi V_w)(s)$: the TD fixed point $\square$

One-hot: the two conditions become $V(s) = V^\pi(s)$ and $V(s) = (\mathcal{T}^\pi V)(s)$ for every state; both give V^π .

Example: the TD fixed point in the layer class. Four groups, four equations: the backup of the fit minus the fit, summed over the group with weights μ .

$$\text{bottom: } \frac{1}{4} [(0 - w_2) + (0 - w_2) + (0 - w_2) + (4 - w_2)] = 0 \Rightarrow w_2 = 1$$

the backup at a bottom state is its last pay: 0, 0, 0, 4; four states, $\mu = \frac{1}{4}$ each

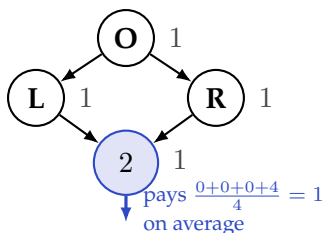
$$\text{L: } \frac{1}{2} [(0 + \gamma w_2) - w_L] = 0 \Rightarrow w_L = \gamma w_2 = 1 \quad \text{the backup at L reads the shared number; } \gamma = 1$$

$$\text{R: } \frac{1}{2} [(0 + \gamma w_2) - w_R] = 0 \Rightarrow w_R = 1 \quad \text{the same number, though R is worth 2}$$

$$\text{O: } 1 \cdot [\gamma \frac{1}{2} (w_L + w_R) - w_O] = 0 \Rightarrow w_O = 1$$

$$V_{w_{\text{TD}}} = (1; 1, 1; 1, 1, 1, 1) \quad \text{against} \quad V_{w_{\text{LS}}} = (1; 0, 2; 1, 1, 1, 1)$$

solved bottom-up, no linear system: no group holds a state and its successor



The lumped tree: one bottom state, entered from L and R alike, paying 1 on average; its value, 1 everywhere, is the TD fixed point.

	O	L	R	LL, LR, RL	RR	$\overline{VE}$
truth V^π	1	0	2	0	4	0
best fit, $w_{LS} = (1, 0, 2, 1)$	1	0	2	1	1	3
its error	0	0	0	+1	-3	
TD fixed point, $w_{TD} = (1, 1, 1, 1)$	1	1	1	1	1	4
its error	0	+1	-1	+1	-3	

Two destinations in one class. The bottom layer costs both 3; TD also loses L and R, which have their own numbers: its target at L is the shared number, which carries the prize of RR. The target of Monte Carlo at L is the return from L: it never passes through the class.

The projected Bellman equation (stated) (Tsitsiklis and Van Roy, 1997), (Sutton and Barto, 2018, Section 9.4).

$$\Pi V := V_w, \quad w = \operatorname{argmin}_w \sum_s \mu(s) (V(s) - V_w(s))^2$$

the best fit to any table V : the point of the plane nearest to V under μ

groups: $(\Pi V)(s)$ = the μ -weighted mean of V over the group of s

on the tree: replace the bottom layer by its mean, keep O, L, R

Monte Carlo stops at $V_w = \Pi V^\pi$

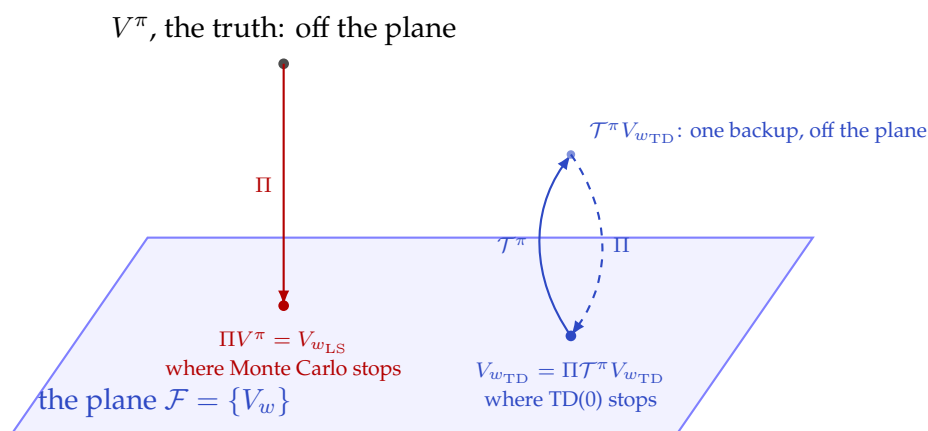
project the truth once

TD(0) stops at $V_w = \Pi \mathcal{T}^\pi V_w$

the fixed-point equations above, read with $V = \mathcal{T}^\pi V_w$: back up the fit, project, land on the same fit

$$\mathcal{T}^\pi V_{w_{TD}} = (1; 1, 1; 0, 0, 0, 4) \xrightarrow{\Pi} (1; 1, 1; 1, 1, 1, 1) = V_{w_{TD}}$$

on the tree: the backup leaves the plane (the bottom layer is not constant); its projection comes back



Two stops on one plane: Monte Carlo drops the truth onto it; TD backs the fit up, leaves the plane, and is projected back onto itself.

Theorem (stated): on-policy linear TD(0) converges, and how far it can land from the best fit (Tsitsiklis and Van Roy, 1997), (Sutton and Barto, 2018, Section 9.4).

$$\mathbb{E}[\delta_t \phi(S_t)] = b - Aw$$

linear in w : the fixed point solves $Aw_{\text{TD}} = b$; the mean update is $w \leftarrow w + \alpha(b - Aw)$

$$A = \sum_s \mu(s) \phi(s) (\phi(s) - \gamma \mathbb{E}[\phi(S_{t+1}) | S_t = s])^\top, \quad b = \sum_s \mu(s) r^\pi(s) \phi(s)$$

the feature of s against the mean feature one step on, and the mean pay, both weighted by μ

layer class: $A = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad b = (0, 0, 0, 1), \quad A(1, 1, 1, 1)^\top = b$

triangular: no group holds a state and its successor; back-substitution is the backward induction above

on-policy (S_t weighted by μ), ϕ linearly independent, $\sum \alpha = \infty$, $\sum \alpha^2 < \infty$

$\Rightarrow w_t \rightarrow w_{\text{TD}}$ with probability 1

under μ the matrix A shrinks every direction (positive definite): the whole reason

episodes that end with probability 1: covered too, with a bound of their own (stated)

Bertsekas and Tsitsiklis (1996)

$$\gamma < 1: \quad \overline{\text{VE}}(w_{\text{TD}}) \leq \frac{1}{1-\gamma} \min_w \overline{\text{VE}}(w)$$

at most ten times the best fit's error at $\gamma = 0.9$; on the tree, $\gamma = 1$, the loss is 4 against 3

Example: the same tree with a discount. $\gamma = \frac{1}{2}$, the prize 32; the layer class.

$$V^\pi = (1; 0, 4; 0, 0, 0, 16) \quad 16 = \frac{1}{2} \cdot 32, \text{ then } \gamma \cdot \frac{1}{2} \text{ per layer: } 4, 1$$

$$\text{best fit: } w_2 = \frac{0+0+0+16}{4} = 4, \quad V_{w_{\text{LS}}} = (1; 0, 4; 4, 4, 4, 4) \quad \overline{\text{VE}} = \frac{1}{4}(16 + 16 + 16 + 144) = 48$$

$$\text{TD: } w_2 = 4, \quad w_L = \gamma w_2 = 2, \quad w_R = 2, \quad w_O = \gamma \frac{1}{2}(w_L + w_R) = 1$$

the discount enters the fixed point where it enters the backup: once per layer

$$V_{w_{\text{TD}}} = (1; 2, 2; 4, 4, 4, 4), \quad \overline{\text{VE}} = 48 + \frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 4 = 52 \leq \frac{1}{1-\gamma} \cdot 48 = 96$$

the bound at $\gamma = \frac{1}{2}$, checked

- Monte Carlo has no such condition and no such factor: gradient descent on $\overline{\text{VE}}$ lands on the best fit for any weighting it is given.
- TD leans on the weighting: feed the states in another proportion, or from another policy's episodes, and A need not shrink every direction, so w can run away (stated) (Baird, 1995). A table never does.
- The loss is $\frac{4}{3}$ here and can reach the bound (stated): bootstrapping through a class without the truth. The dead-end class holds it: both stops are the truth.

Reading (optional): on a fixed record, and the full gradient. Nothing below depends on this box. Hold a record of N transitions and replay it. Batch semi-gradient TD(0) stops where $\sum_i \phi(s_i)(r_i + \gamma V_w(s'_i) - V_w(s_i)) = 0$, the record's own d equations: solving them directly is least-squares TD (Bradtke and Barto, 1996); batch gradient Monte Carlo stops at least squares on the record's returns. With one-hot features the first is certainty equivalence in the record's estimated chain. Why not differentiate the target too? The full gradient of $(R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t))^2$ descends a different objective, $\sum_s \mu(s)((\mathcal{T}^\pi V_w)(s) - V_w(s))^2 + \gamma^2 \sum_s \mu(s) \text{Var}[V_w(S_{t+1}) \mid S_t = s]$, which penalizes a fit whose next-state value varies: where the next state is a coin, as on the tree, it stops at yet another point (Baird, 1995). The semi-gradient keeps the target fixed, as a table did. The theorem above needs a linear V_w : Tsitsiklis and Van Roy (1997) also give a small chain and a nonlinear V_w on which on-policy semi-gradient TD(0) spirals away (stated). For gradient Monte Carlo, the mean update is $-\frac{1}{2}\nabla\overline{\text{VE}}$ instead.

By Computer: Two Destinations

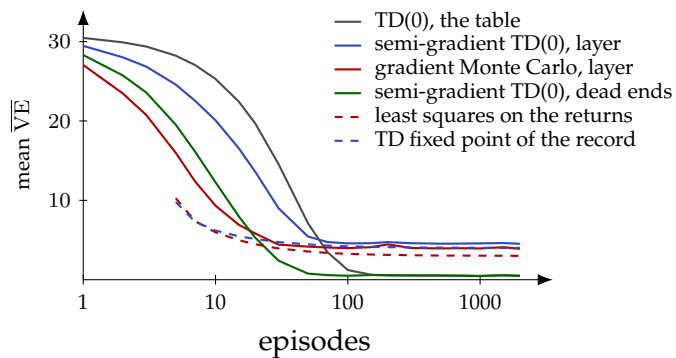
The experiment.

1. 200 runs of 2,000 episodes from O, every number started at 4 (from 0 the dead ends are right by accident); after n episodes each learner holds a fit V_n .
2. Reported: the mean of $\overline{\text{VE}}(V_n) = \sum_s \mu(s)(V_n(s) - V^\pi(s))^2$ over the runs at $n = 10, 20, 50, 100, 500, 2,000$; at the start it is 31.

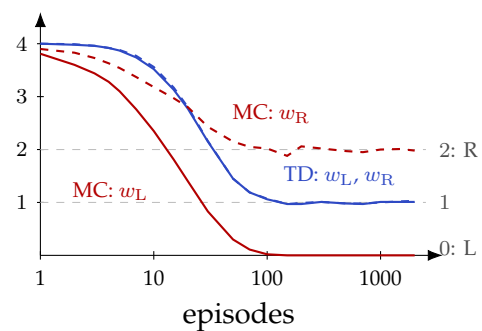
Method	What V_n is
TD(0) on the table, $\alpha = 0.1$	seven numbers, one pass over the first n episodes, from 4
semi-gradient TD(0), the layer class, $\alpha = 0.1$	the box above, one pass, from $w = (4, 4, 4, 4)$
gradient Monte Carlo, the layer class, $\alpha = 0.1$	one step per state of the episode, when the episode ends, from $(4, 4, 4, 4)$
semi-gradient TD(0), the dead-end class, $\alpha = 0.1$	the box above with the other four numbers, from $(4, 4, 4, 4)$
least squares on the returns, the layer class	$\text{argmin}_w \sum_u (G_u - V_w(S_u))^2$ over the returns of the first n episodes: where gradient Monte Carlo is headed
the TD fixed point of the record, the layer class	w solving $\sum_u \phi(S_u)(R_{u+1} + V_w(S_{u+1}) - V_w(S_u)) = 0$ over the first n episodes: where semi-gradient TD(0) is headed (reading box)

mean $\overline{VE}$ over 200 runs after $n =$	10	20	50	100	500	2,000
TD(0) on the table, $\alpha = 0.1$	25.29	19.67	7.11	1.22	0.53	0.49
semi-gradient TD(0), the layer class, $\alpha = 0.1$	20.08	13.58	5.44	4.58	4.54	4.53
gradient Monte Carlo, the layer class, $\alpha = 0.1$	9.33	5.82	4.17	3.98	3.94	3.89
semi-gradient TD(0), the dead-end class, $\alpha = 0.1$	12.27	5.28	0.77	0.50	0.53	0.49
least squares on the returns, the layer class	5.97	4.47	3.56	3.26	3.04	3.01
the TD fixed point of the record, the layer class	6.20	5.13	4.45	4.21	4.04	4.01

Seed 639. Destinations: 0 for the table and the dead-end class, 3 and 4 for the layer class under Monte Carlo and TD; the two bottom rows reach 3.01 and 4.01.



The table as curves, the episode axis logarithmic: blue heads for 4, red for 3, gray and green for 0.



The layer class, $\alpha = 0.1$, mean over the runs: w_L (solid), w_R (dashed) under TD (blue) and Monte Carlo (red); Monte Carlo parts them to 0 and 2, TD holds them at 1.

- Early, every class beats the table (episode 20: 5.28, 5.82, 13.58 against 19.67): four numbers, each moved every episode.
- Late, the floors: 0.49 for the table and the dead-end class; 3.9 and 4.5 for the layer class under Monte Carlo and TD, the two destinations plus the step's noise.

Summary

Function approximation for prediction, on one board

$V_w(s) = w^\top \phi(s)$, $\nabla_w V_w = \phi$, $\mathcal{F} = \{V_w\}$; $\phi = e_s$: the table, $\phi = e_{g(s)}$: a group
the tree, $\gamma = 1$: four numbers, the layer class or the dead-end class

$\overline{VE}(w) = \sum_s \mu(s)(V^\pi(s) - V_w(s))^2$, μ the visits per episode; $\Pi V^\pi = V_{w_{LS}}$
 $w_{LS} = \operatorname{argmin} \overline{VE}$; layer class: (1; 0, 2; 1, 1, 1, 1), $\overline{VE} = 3$; dead-end class: the truth

Monte Carlo: $w \leftarrow w + \alpha(G_t - V_w(S_t))\nabla_w V_w(S_t)$, mean step $-\frac{1}{2}\nabla \overline{VE} \Rightarrow w_t \rightarrow w_{LS}$
least squares on the returns (stated); two episodes: (2; 0, 4; 2, 2, 2, 2)

TD(0): $w \leftarrow w + \alpha \delta_t \nabla_w V_w(S_t)$, $\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$
the target frozen; any differentiable V_w ; two episodes: (0; 2, 0; 2, 2, 2, 2), the table (0; 0, 0; 0, 0, 0, 4)

$\sum_s \mu \phi (V^\pi - V_w) = 0 (w_{LS})$ against $\sum_s \mu \phi (\mathcal{T}^\pi V_w - V_w) = 0 (w_{TD})$
tree: $w_{TD} = (1, 1, 1, 1)$, $\overline{VE} 4$ against 3; on-policy, linear, Robbins–Monro: $w_t \rightarrow w_{TD}$ (stated)

by computer from 4: $n = 20$: 19.67/13.58/5.82/5.28; $n = 2,000$: 0.49/4.53/3.89/0.49
table / layer TD / layer Monte Carlo / dead-end TD: two floors, 4 and 3, and two zeros

References

- Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.
- Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
- Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1–3):33–57, 1996.
- Emma Brunskill. CS 234: Reinforcement learning. Stanford University course materials, 2026. URL <https://web.stanford.edu/class/cs234/>. Winter 2026 offering.
- Nan Jiang. CS 443: Reinforcement learning. University of Illinois Urbana-Champaign course materials, 2026. URL <https://nanjiang.cs.illinois.edu/cs443/>. Spring 2026 offering.
- Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.
- Wen Sun. CS 4/5789: Introduction to reinforcement learning. Cornell University course materials, 2025. URL https://wensun.github.io/CS4789_spring_2025.html. Spring 2025 offering.

- Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.
- Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010. URL <https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf>.
- Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.
- John Tromp and Gunnar Farneback. Combinatorics of Go. *Computers and Games (CG 2006), Lecture Notes in Computer Science*, 4630:84–99, 2007.
- John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.