Lecture -06

- Dynamic Programming
- Monte Carlo
- Temporal Difference (TD) learning.
- n-step TD methods.

Value Function approximation

Tabular methods

→ large state space — memory — sig. time & data.

→ States rarely repeat.

→ Generalization

Supervised learning.

(x_1, y_1)

(x_2, y_2)

(x_n, y_n)

$f : x \to y$

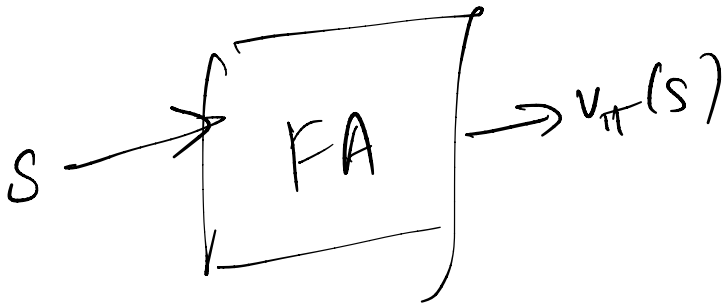
→ linear models

→ Non linear models (neural nets)

goal: estimate value fns.

① large state spaces.

② need for generalization to unseen states.



① learning incrementally.

② Value FA → Nonstationary target fn.

forget: $R_{t+1} + \gamma v_\pi(S_{t+1})$

Value fn. approximation :-

$V_\pi(s)$ – value fn.

$$\tilde{V}(s;\omega) \approx V_\pi(s)$$

weights.

approximation.

$\omega \in \mathbb{R}^d$

$d << |S|$

linear model: 's'

$$x(s) = (x_1(s), x_2(s) \ldots x_d(s))$$

representation of the state.

$$\tilde{V}(s;\omega) = \omega^T x(s)$$

$$= \sum_{i=1}^{d} \omega_i x_i(s)$$

State	Value
s_1	
s_2	
s_3	
⋮	
s_{100}	

← tabular approach.
no generalization.

State → target update.

$S \longmapsto u$

Monte Carlo $\quad S_t \longmapsto G_t$

1-step TD $\quad S_t \longmapsto R_{t+1} + \gamma \hat{v}(S_{t+1}; w_t)$

n-step TD $\quad S_t \longmapsto G_{t:t+n}$

DP $\quad s \longmapsto \mathbb{E}_\pi \left[R_{t+1} + \gamma \hat{v}(S_{t+1}; w_t) \mid S_t = s \right]$

(S_t, G_t)

FA: ~~□~~ $\quad d \ll |S|$

$\mu(s) \quad \mu(s) \geq 0 \quad , \sum_s \mu(s) = 1$

The prediction objective.

Mean squared error.

$$\overline{VE}(w) = \sum_{s \in S} \mu(s) \left(v_\pi(s) - \hat{v}(s, w) \right)^2$$

$$VE(w) = \frac{1}{|S|} \sum_{s \in S} \left(v_\pi(s) - \hat{v}(s; w) \right)^2$$

$$w^* = \operatorname*{argmin}_w VE(w)$$

$\mu(s)$

$$\overline{VE}(w) = \sum_{s \in S} \mu(s) \left(v_\pi(s) - \hat{v}(s; w) \right)^2$$

mean squared error

Policy gradient methods

Square root of $\overline{VE}$

$$w^* = \operatorname*{argmin}_w \overline{VE}(w)$$

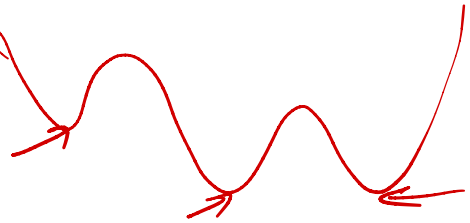
global optimum w^*

$$\overline{VE}(w^*) \leq \overline{VE}(w) \quad \text{for all possible } w$$

local optimum

$$\overline{VE}(w^*) \leq \overline{VE}(w)$$

for all w in some neighborhood of w^*.



On-policy prediction with approximation :-

Stochastic gradient descent :-

$$w = (w_1, w_2 \dots w_d)^T$$

$$\hat{v}(s; w) = w^T x(s)$$

$$w_0, w_1, w_2, w_3 \dots$$

$$S_t \longmapsto v_\pi(S_t)$$

step size

SGD :

$$w_{t+1} = w_t - \frac{1}{2}\alpha \frac{\partial}{\partial w}\left[v_\pi(S_t) - \hat{v}(S_t; w_t)\right]^2$$

$$w_{t+1} = w_t + \alpha\left[v_\pi(S_t) - \hat{v}(S_t; w_t)\right]\frac{\partial}{\partial w}\hat{v}(S_t; w_t)$$

$$w_{t+1} = w_t + \alpha\left[U_t - \hat{v}(S_t; w_t)\right]\frac{\partial}{\partial w}\hat{v}(S_t; w_t)$$

if U_t is unbiased,

$$\mathbb{E}[U_t \mid S_t = s] = V_\pi(s_t)$$

$$U_t = G_t$$

Prediction:

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G_t - \hat{v}(S_t,\mathbf{w})\big]\nabla\hat{v}(S_t,\mathbf{w})$

$\hat{v}(s,w) = w^T x(s)$

n-step return : $G_{t:t+n}$

DP : $\sum_{a,s',r} \pi(a|s_t)\, p(s',r|s_t,a)\,[r + \gamma \hat{V}(s';w_t)]$

1 step-TD : $R_{t+1} + \gamma \hat{V}(S_{t+1};w)$

error = $\left(\underbrace{R_{t+1} + \gamma \hat{V}(S_{t+1};w)}_{\text{Constant}} - \hat{V}(S_t;w)\right)^2$

$\frac{\partial \text{error}}{\partial w}$

Semi-gradient

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

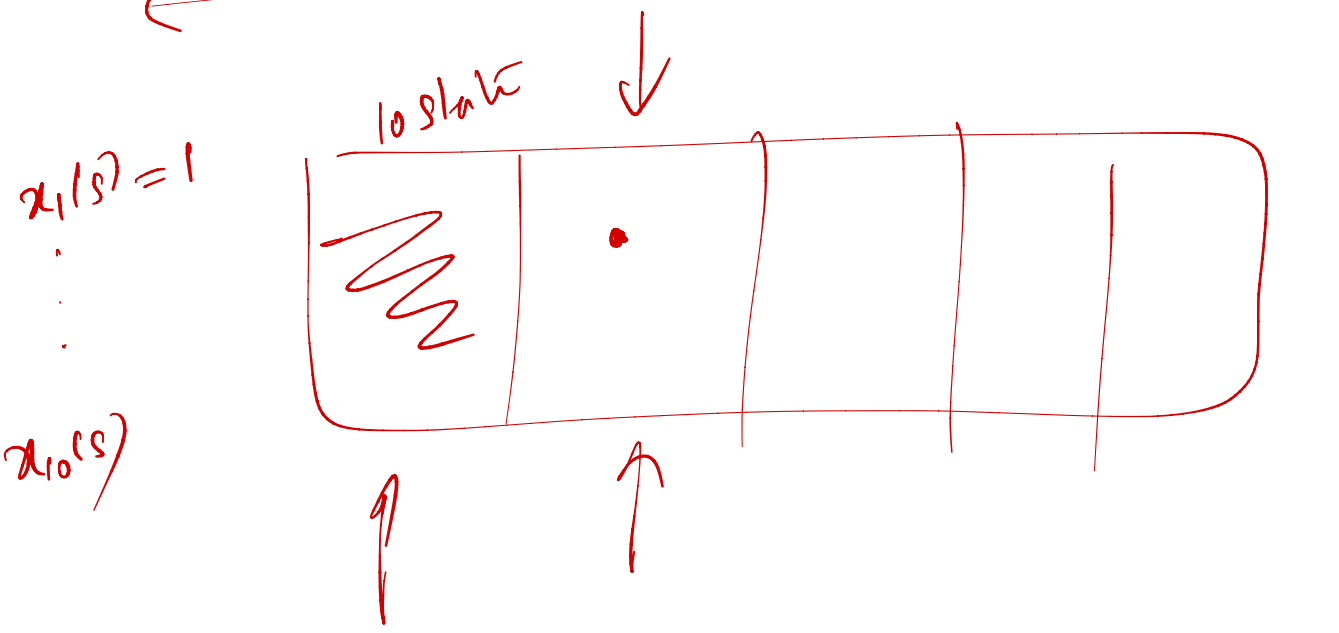
Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S is terminal

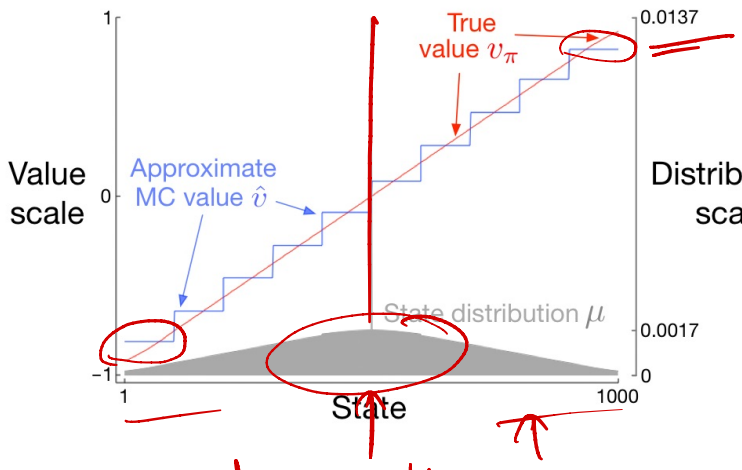
tabular methods

State aggregation

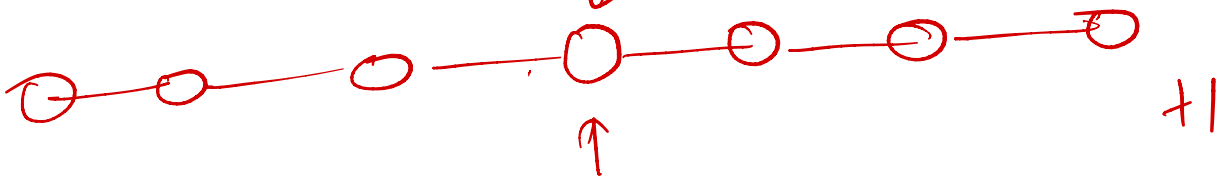
function approx



1.37-1.



1000-state random walk



Linear methods:

$$x(s) = \begin{pmatrix} x_1(s), x_2(s) \cdots x_d(s) \end{pmatrix}^T$$

$$\hat{v}(s;w) = w^T x(s) = \sum_{i=1}^{d} w_i x_i(s)$$

$$\frac{\partial}{\partial w} \hat{v}(s;w) = x(s)$$

SGD update:

$$w_{t+1} = w_t + \alpha \left[U_t - \hat{v}(s_t;w_t) \right] x(s_t)$$

Semi-gradient TD :

Continuing Case :

$$w_{t+1} = w_t + \alpha \left(R_{t+1} + \gamma w_t^T x_{t+1} - w_t^T x_t \right) x_t$$

$$= w_t + \alpha \left(\underbrace{R_{t+1} x_t}_{b} - \underbrace{x_t (x_t - \gamma x_{t+1})^T w_t}_{A} \right)$$

$$\mathbb{E}[w_{t+1} \mid w_t] = w_t + \alpha(b - A w_t)$$

where $b = \mathbb{E}[R_{t+1} x_t] \in \mathbb{R}^d$

$$A = \mathbb{E}\left[x_t (x_t - \gamma x_{t+1})^T \right] \in \mathbb{R}^{d \times d}.$$

at Convergence w_{TD}

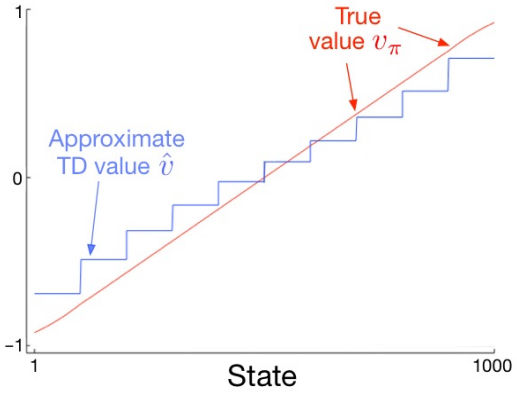
$$b - A w_{TD} = 0$$

$$b = A w_{TD}$$

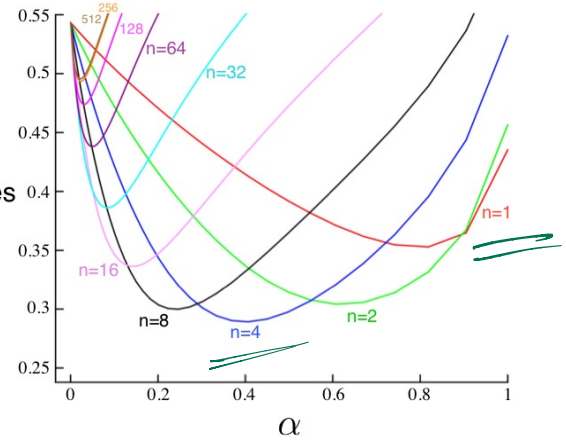
$$w_{TD} = A^{-1} b.$$

TD fixed point

$$\overline{VE}(w_{TD}) \leq \frac{1}{1-\gamma} \min_{w} \overline{VE}(w)$$



Average RMS error over 1000 states and first 10 episodes



$$w_{TD} = A^{-1} b$$

$$A = \mathbb{E}\left[x_t (x_t - \gamma x_{t+1})^\top \right]$$

$$b = \mathbb{E}\left[R_{t+1} x_t \right]$$

Compute estimate of A and b.

Least-Squares TD (LSTD)

$$\hat{A}_t = \sum_{k=0}^{t-1} x_k (x_k - \gamma x_{k+1})^\top + \epsilon I$$

$\epsilon \to 0$

$$\hat{b}_t = \sum_{k=0}^{t-1} R_{k+1} x_k$$

$$w_t = \hat{A}_t^{-1} \hat{b}_t$$

Semi-gradient TD

- iterative algorithm
  esh. val.

$O(d)$

LSTD.

$\rightarrow$ esh. A, b.

Comp. $A_t : O(d^2)$

$A_t^{-1} : O(d^3)$

$O(d^2)$

$$\hat{A}_t^{-1} = \left(\hat{A}_{t-1} + x_{t-1}(x_{t-1} - \gamma x_t)^T \right)^{-1}$$

$$= \hat{A}_{t-1}^{-1} - \frac{\hat{A}_{t-1}^{-1} x_{t-1}(x_{t-1} - \gamma x_t)^T \hat{A}_{t-1}^{-1}}{1 + (x_{t-1} - \gamma x_t)^T \hat{A}_{t-1}^{-1} x_{t-1}}$$

$\hat{A}_0 = \epsilon I$.

Sherman - Morrison formula.

LSTD for estimating $\hat{v} = \mathbf{w}^\top \mathbf{x}(\cdot) \approx v_\pi$ ($O(d^2)$ version)

Input: feature representation $\mathbf{x} : \mathcal{S}^+ \to \mathbb{R}^d$ such that $\mathbf{x}(terminal) = \mathbf{0}$
Algorithm parameter: small $\varepsilon > 0$

$\widehat{\mathbf{A}^{-1}} \leftarrow \varepsilon^{-1}\mathbf{I}$ — A $d \times d$ matrix
$\mathbf{b} \leftarrow \mathbf{0}$ — A d-dimensional vector
Loop for each episode:
  Initialize S; $\mathbf{x} \leftarrow \mathbf{x}(S)$
  Loop for each step of episode:
    Choose and take action $A \sim \pi(\cdot|S)$, observe R, S'; $\mathbf{x}' \leftarrow \mathbf{x}(S')$
    $\mathbf{v} \leftarrow \widehat{\mathbf{A}^{-1}}^\top (\mathbf{x} - \gamma \mathbf{x}')$
    $\widehat{\mathbf{A}^{-1}} \leftarrow \widehat{\mathbf{A}^{-1}} - \big(\widehat{\mathbf{A}^{-1}}\mathbf{x}\big)\mathbf{v}^\top / \big(1 + \mathbf{v}^\top \mathbf{x}\big)$
    $\widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} + R\mathbf{x}$
    $\mathbf{w} \leftarrow \widehat{\mathbf{A}^{-1}}\widehat{\mathbf{b}}$
    $S \leftarrow S'$; $\mathbf{x} \leftarrow \mathbf{x}'$
  until S' is terminal

Representation of states :-

Linear models :

$w^T x$

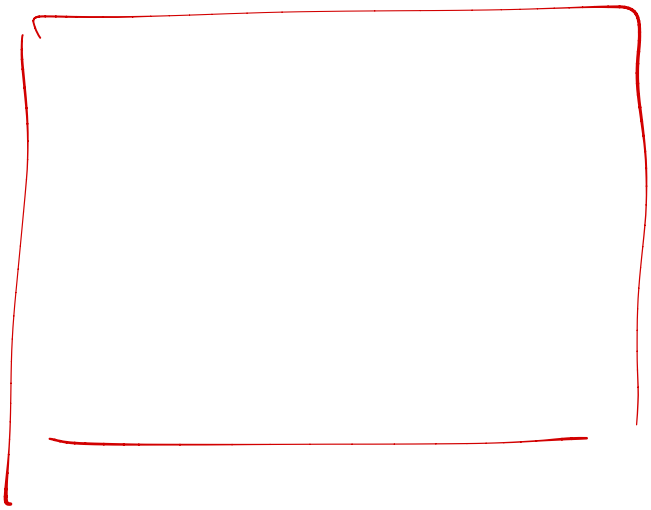
Polynomials

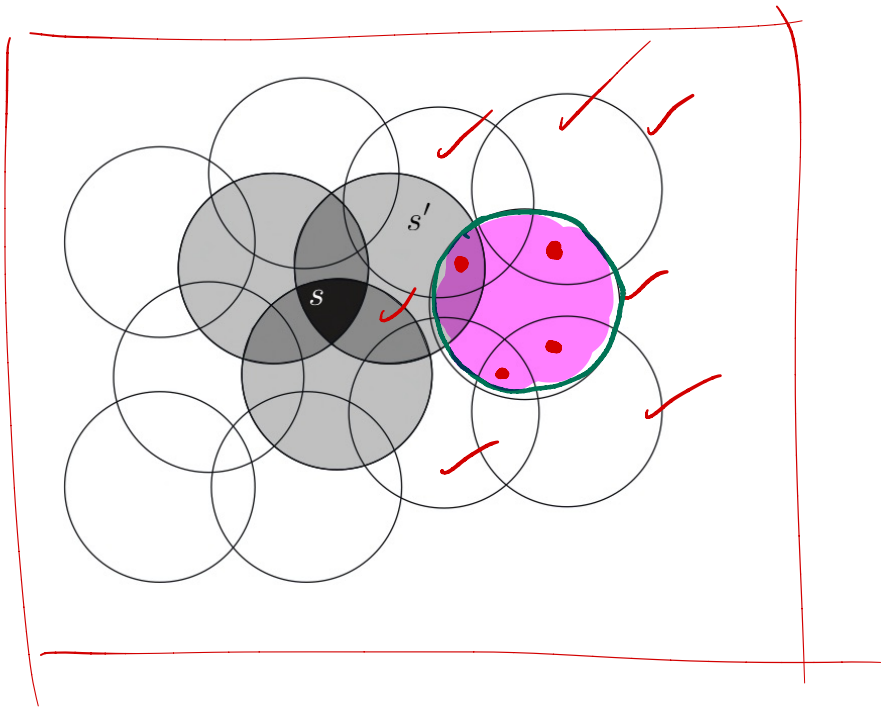
(S_1, S_2)

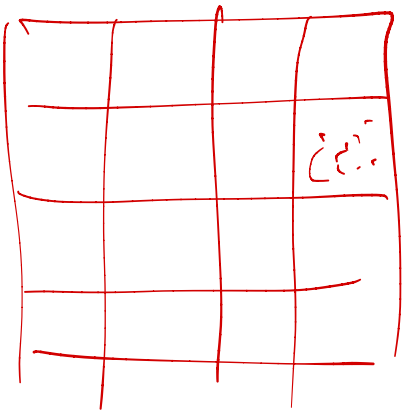
$(1, S_1, S_2, S_1 S_2)$

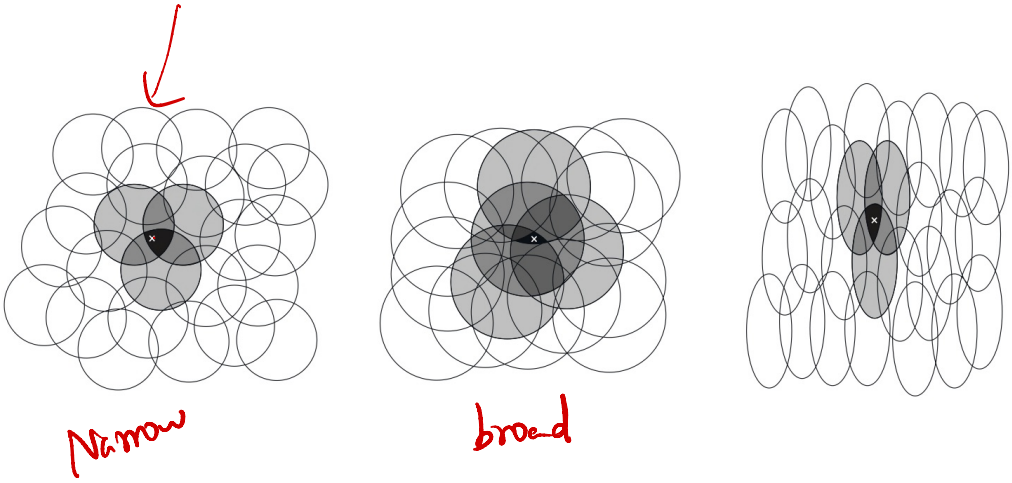
$(1, S_1, S_2, S_1^2, S_2^2, S_1 S_2)$

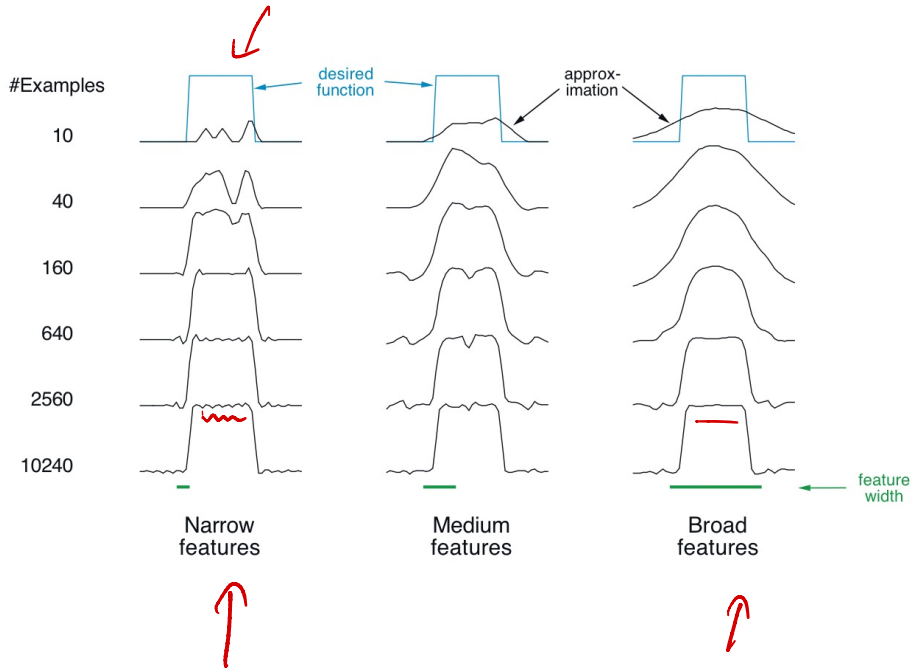
Coarse Coding:



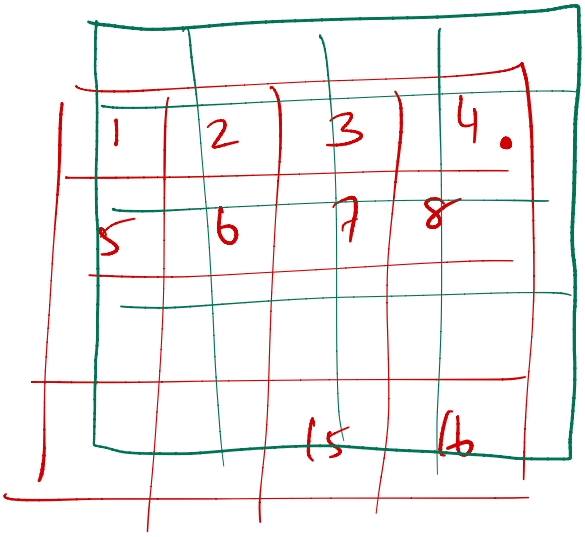

s'
s

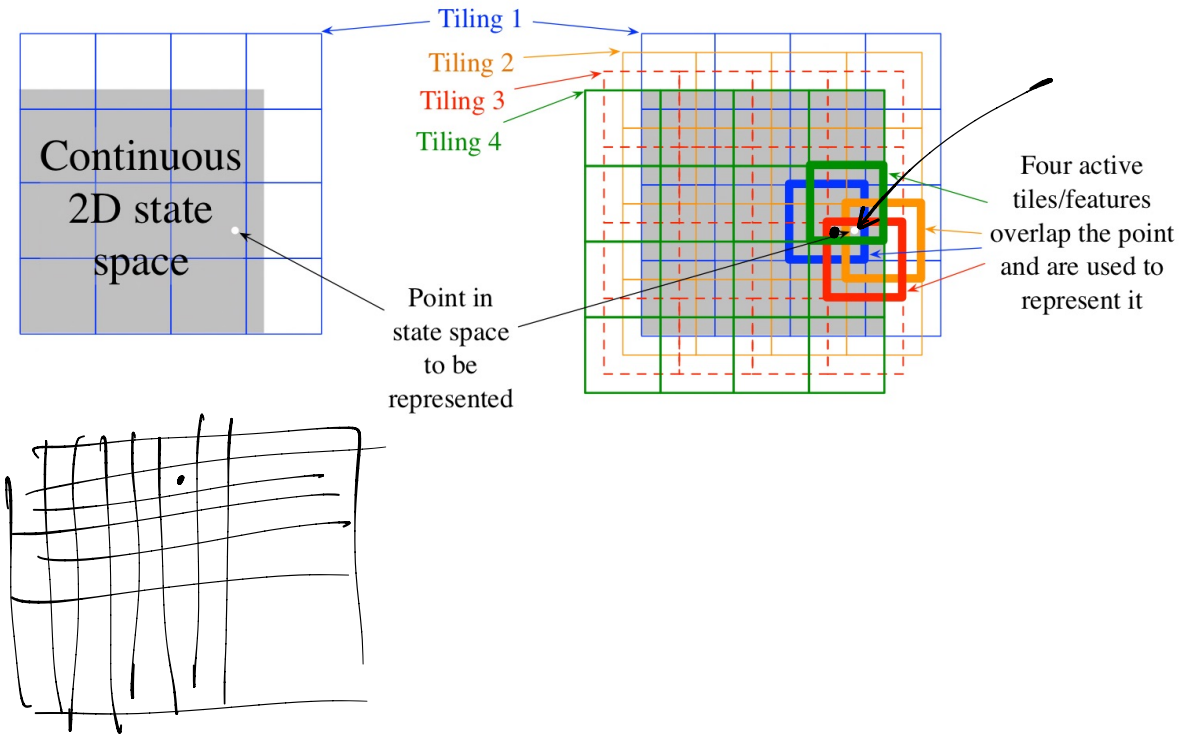


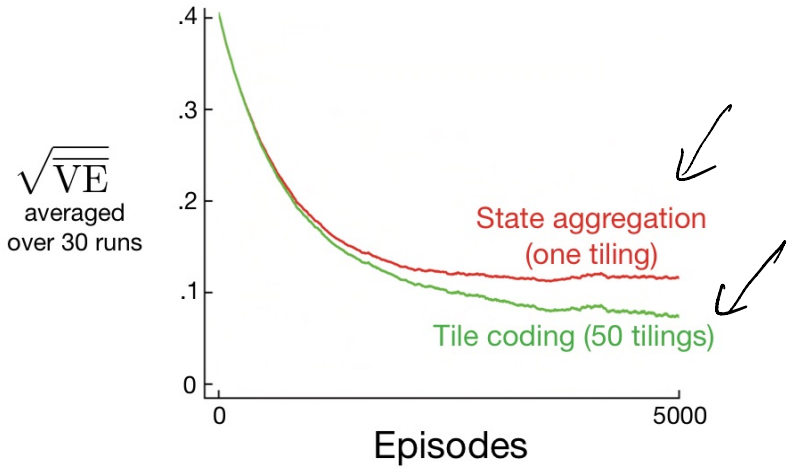

Narrow
broad


#Examples
10
40
160
640
2560
10240
desired function
approx-imation
feature width
Narrow features
Medium features
Broad features

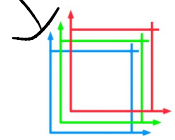
Tile-coding :

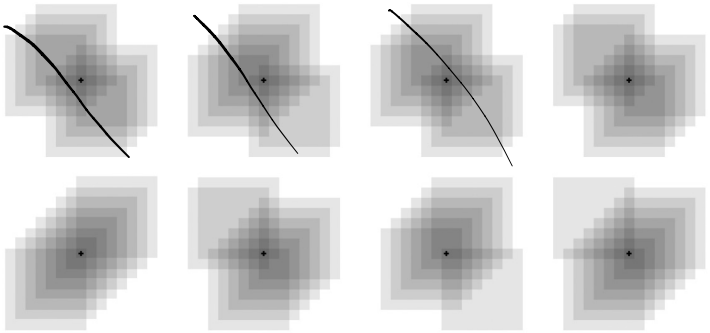

1
2
3
4
5
6
7
8
15
16


Tiling 1
Tiling 2
Tiling 3
Tiling 4
Continuous 2D state space
Point in state space to be represented
Four active tiles/features overlap the point and are used to represent it

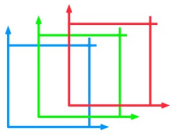

$\sqrt{\overline{\text{VE}}}$
averaged
over 30 runs
.4
.3
.2
.1
0
0
5000
Episodes
State aggregation
(one tiling)
Tile coding (50 tilings)

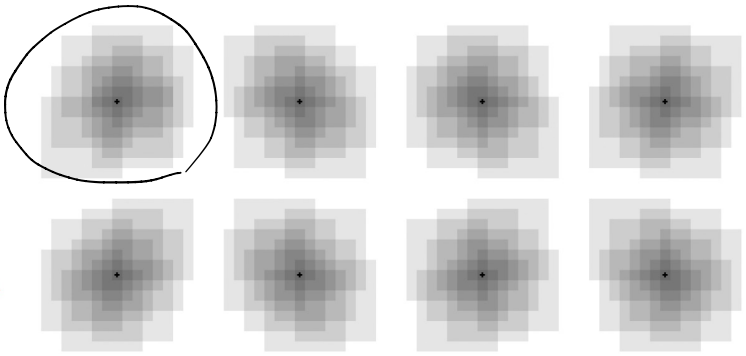
Possible generalizations for uniformly offset tilings





Possible generalizations for asymmetrically offset tilings





On-policy Control with Approximation :-

$$w_{t+1} = w_t + \alpha \left[U_t - \hat{q}(S_t, A_t; w_t) \right] \frac{\partial \hat{q}(S_t, A_t; w_t)}{\partial w}$$

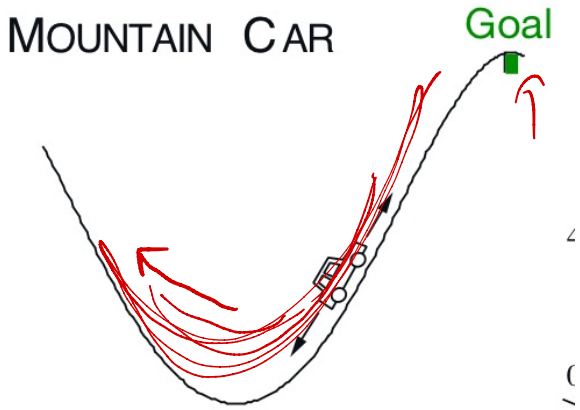
one-step Sarsa :-

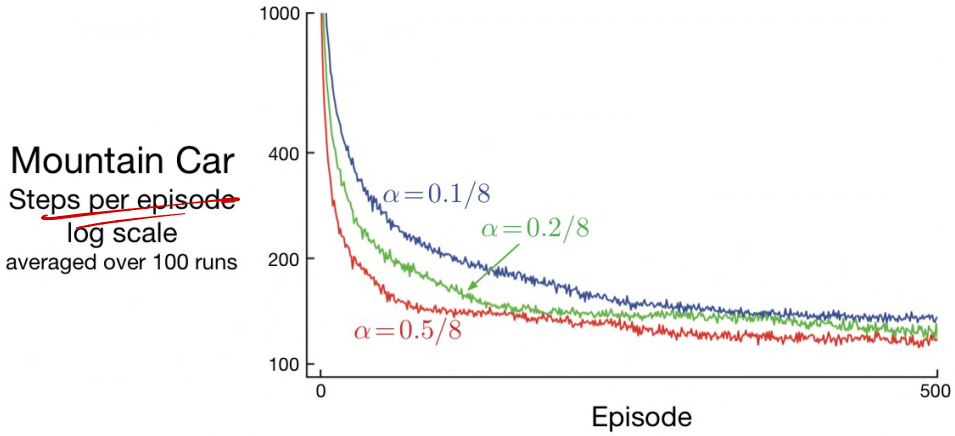
$$w_{t+1} = w_t + \alpha \left[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}; w_t) - \hat{q}(S_t, A_t; w_t) \right] \frac{\partial}{\partial w} \hat{q}(S_t, A_t; w_t)$$

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

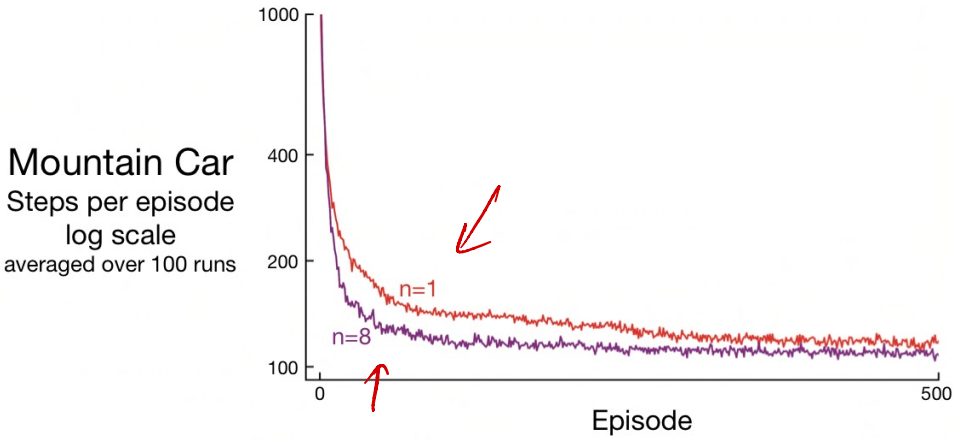

MOUNTAIN CAR
Goal


Mountain Car
Steps per episode
log scale
averaged over 100 runs
1000
400
200
100
0
500
Episode
$\alpha = 0.1/8$
$\alpha = 0.2/8$
$\alpha = 0.5/8$

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_*$ or q_π

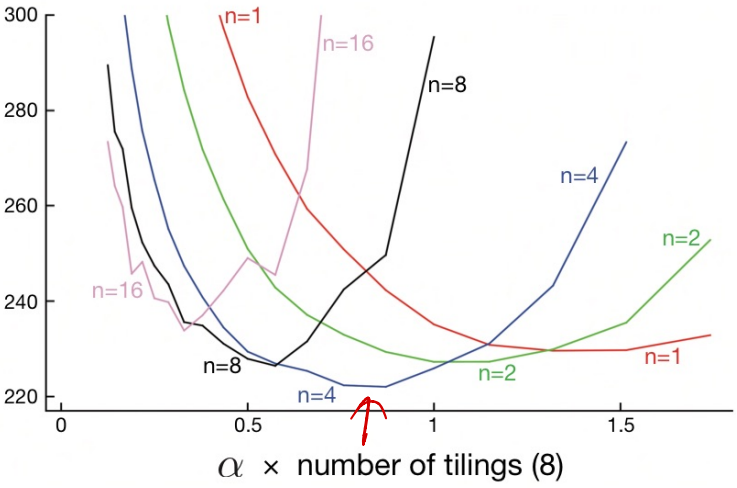
Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
    If $t < T$, then:
      Take action A_t
      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
      If S_{t+1} is terminal, then:
        $T \leftarrow t + 1$
      else:
        Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $(G_{\tau:\tau+n})$
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})\right] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$


Mountain Car
Steps per episode
log scale
averaged over 100 runs
1000
400
200
100
0
500
Episode
n=1
n=8

Mountain Car

Steps per episode
averaged over
first 50 episodes
and 100 runs



Value fn. approximation