Intro to Value function approximation

Sunday, May 10, 2020 4:00 PM

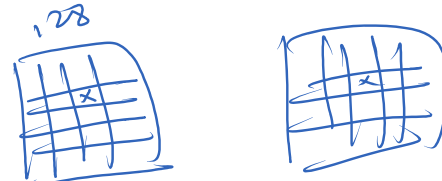
Reinforcement Learning
Value function approximation

High-dimension state space:

10^45

128

128



S_1 S_2

0 - 255

$256^{(128 \times 128)}$

function approximation

$S_1 \rightarrow$	V_1
$S_2 \rightarrow$	V_2

$S_2 \to$	v_2

Supervised-learning perspective for approximation:

$Monte - Carlo\ update\ equation:$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

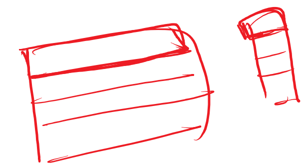
$TD(0)\ update\ equation:$

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

New EA $\leftarrow$ Old EA + α [Target − Old EA]

$X \to Y$

$S_t \to u_t$



$S_1 \to u_1$

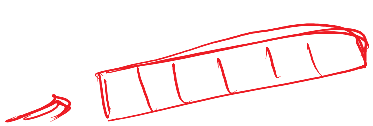
$S_2 \to u_2$

⋮

$S_T \to u_T$

$x \to$ [ ] $\to \hat{y}$

$\to f(x) \to \hat{y}$



$x(S_1) \to u_1 \quad (R)^2$

$x(S_2) \to u_2$

$x(S_3) \to u_3$

⋮

$$x(S_t) \longrightarrow \boxed{\hat{v}(S, w)} \longrightarrow \hat{v}(S_t, w)$$

Reinforcement Learning
Value Error

Value Error

$\mu(s)$

$\sum_{s} \mu(s) = 1$

$(\hat{y}_i - y_i)^2$

$[\hat{v}(s,w) - v_\pi(s)]^2$

Mean squared Value Error is defined as:

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

$w^* \rightarrow VE(w^*) \leq VE(w)$

Gradient based learning

Friday, May 15, 2020 6:45 PM

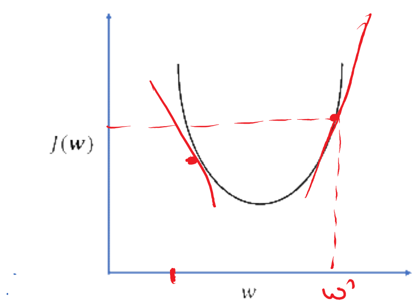
Reinforcement Learning
Gradient Monte-Carlo & Semi-gradient TD

Stochastic Gradient Descent

$$J(w) = \sum\left[y_i - f(w)\right]^2$$

$$\nabla J(w) = 2 * \sum\left[y_i - f(w)\right]\nabla f(w)$$

$$w \leftarrow w - \alpha \nabla J(w)$$



$$w' \geq w \quad \text{if} \quad \nabla > 0$$

Stochastic Gradient Descent on Value Error

$\hat{v}(s, w)$

S_t $\quad U_t$

$\mu(s)$

SGD weight update for Value Error:

$$w_{t+1} = w_t - \frac{1}{2}\alpha\nabla[v_\pi(S_t) - \hat{v}(S_t, w_t)]^2$$
$$= w_t + \alpha[v_\pi(S_t) - \hat{v}(S_t, w_t)]\nabla\hat{v}(S_t, w_t)$$

U_t

$$\nabla\hat{v}(S_t, w_t) = \left(\frac{\partial\nabla\hat{v}(S_t, w_t)}{\partial w_t^{(1)}}, \frac{\partial\nabla\hat{v}(S_t, w_t)}{\partial w_t^{(2)}}, \frac{\partial\nabla\hat{v}(S_t, w_t)}{\partial w_t^{(3)}}, \dots, \frac{\partial\nabla\hat{v}(S_t, w_t)}{\partial w_t^{(d)}}\right)$$

$$\begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(d)} \end{bmatrix}_{t+1} = \begin{bmatrix} w^{(1)} \\ \vdots \\ w^{(d)} \end{bmatrix}_t + c \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

T

Gradient Monte-Carlo

Gradient Monte − carlo weight update rule:

$$w_{t+1} = w_t + \alpha[G_t - \hat{v}(S_t, w_t)]\nabla\hat{v}(S_t, w_t)$$

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

Semi-Gradient TD(0)

Semi Gradient TD(0) *weight update rule*:

$$w_{t+1} = w_t + \alpha[R_{t+1} + \gamma\hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)]\nabla\hat{v}(S_t, w_t)$$

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

Linear Function Approximation

Friday, May 15, 2020 8:16 PM

Reinforcement Learning
Linear Function Approximation

<u>State value function using linear function approximator</u>

$S \longrightarrow x(S)$ $\quad |x(S)| = |w|$

$x(S) = [x_1 \quad x_2 \quad x_3]^T$

$w = [w_1 \quad w_2 \quad w_3]$

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

$\hat{v}(s,w) = (x_1 w_1 + x_2 w_2 + x_3 w_3)$

$\nabla \hat{v}(s,w) = \left[\frac{\partial \nabla \hat{v}(s,w)}{\partial w_1}, \frac{\partial \nabla \hat{v}(s,w)}{\partial w_2}, \frac{\partial \nabla \hat{v}(s,w)}{\partial w_3} \right]$

$$\nabla \hat{v}(s,w) = \boldsymbol{x}(s)$$

$\nabla\hat{v}(s, w) = \boldsymbol{x}(s)$

$\frac{\partial \hat{v}}{\partial w_3}$

$= [x_1 \quad x_2 \quad x_3]$

$= x(s)$

SGD update for linear function approximator:

$$w_{t+1} = w_t + \alpha[U_t - \hat{v}(S_t, w_t)]\boldsymbol{x}(s)$$

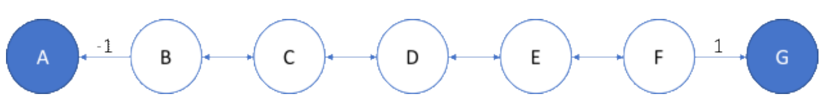
$$VE(w_{TD}) \leq \left(\frac{1}{1-\gamma}\right)\left(VE(w_*)\right)$$

State aggregation

Friday, May 15, 2020 10:53 PM

Reinforcement Learning

State Aggregation



D: $x(D) = [0 \; 0 \; 0 \; 1 \; 0 \; 0 \; 0]$

A: $x(A) = [1 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0]$

B: $x(B) = [0 \; 1 \; 0 \; 0 \; 0 \; 0 \; 0]$

$w = [w_1 \; w_2 \; w_3 \; w_4 \; w_5 \; w_6 \; w_7]$

D: $x(D) \longrightarrow U_t$

SGD update for linear function approximator:

$$w_{t+1} = w_t + \alpha\left[U_t - \hat{v}(S_t, w_t)\right]\boldsymbol{x}(s)$$

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} + \alpha\, C \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

10^8 States

10^6

(1 - 100) : 1

101 - 200 : 2

10^6

$[1\ 0\ 0\ 0\ 0\ \ldots\ 0]$

$[0\ 1\ 0\ 0\ \ldots\ 0]$

1000 state random-walk MDP with state aggregation:

- States are numbered form 1 to 1000, from left to right.
- All episode begin near the center on state 500.
- States 1 and 1000 are terminal states, and it fetches a reward of -1 and 1 on transitioning to them, respectively.
- Policy:
  - Agent can transition from current state to one of 100 states in left or rights, all with equal probability. So there are 200 possible actions. At edges, if an action goes beyond 1 or 1000, then the agent instead transition to the terminal states.

400 499 500 501 600

1 - 100 → 1 :

$[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

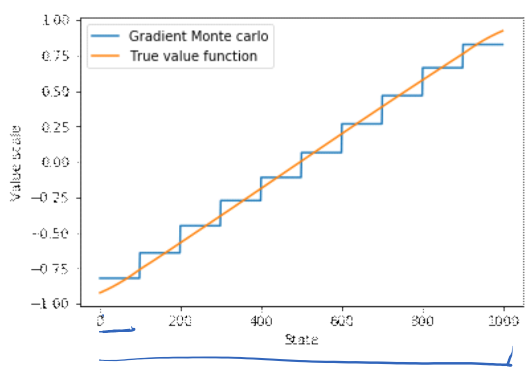
1 - 100 → 1 [1 0 0 0 0 0 0 0 0 0]

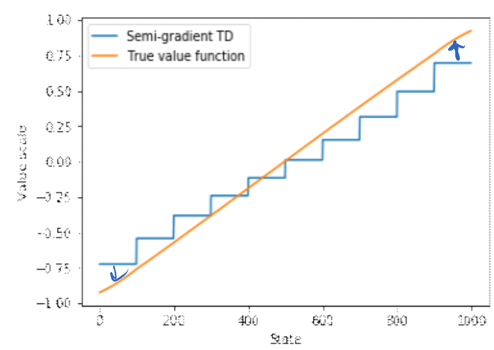
101 → 200 → 2 [0 1 0 0 0 0 0 0 0 0]

201 - 300 → 3

⋮

901 - 1000 → 10 [0 0 0 0 0 0 0 0 0 1]





Coarse coding and Tile coding

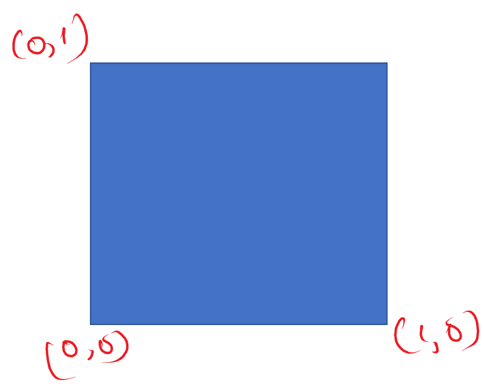
Saturday, May 16, 2020 1:45 AM

Reinforcement Learning

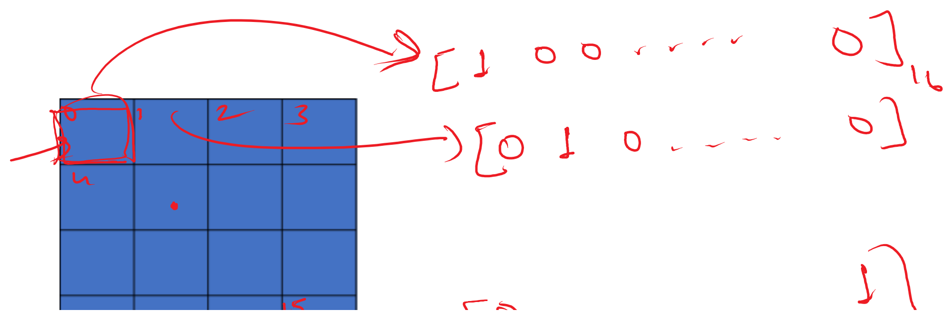
Coarse coding and tile coding

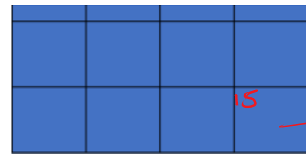
Coarse coding:

(x,y): $x \in [0,1)$, $y \in [0,1]$



(0.007, 0)
(0.002, 0)





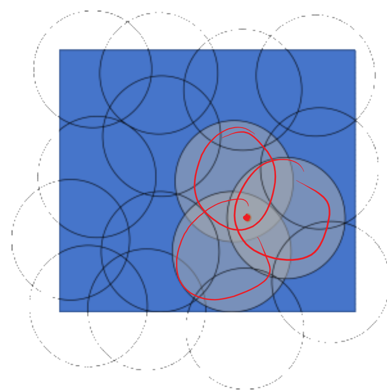
$\to [0 \dots 1]$

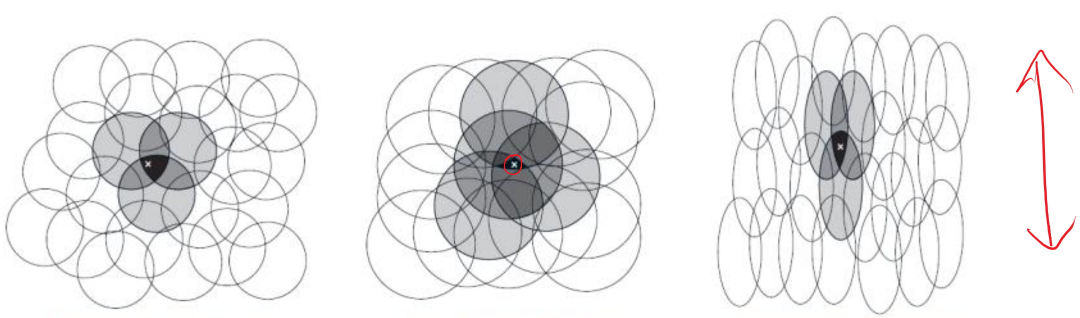
n circles

$[0\ 0\ 1\ 0\ 1\ 0\ 1]_n$

$\downarrow$

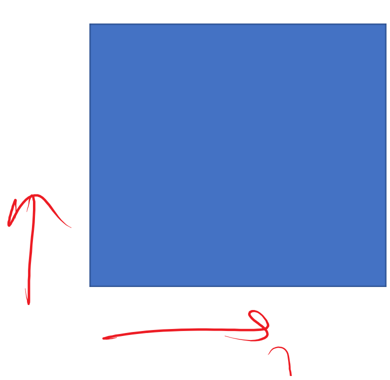
$[w_1\ w_2 \dots w_n]$

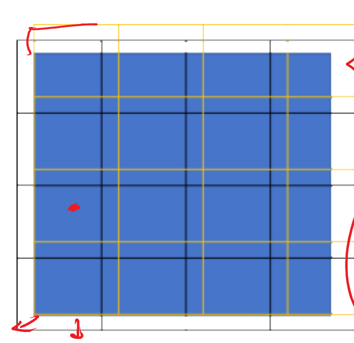




Narrow generalization Broad generalization Asymmetric generalization

Tile coding:





Tiling 1

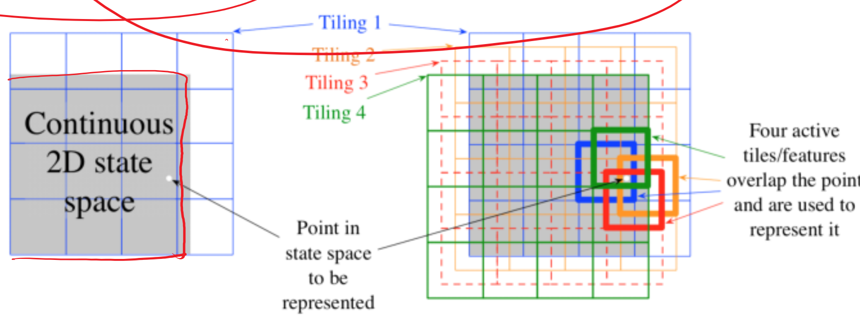
Tiling 2

$[\quad]_{16}$

$]_{11}$

32

16 16



16x4

1000 state random-walk MDP:

- States are numbered form 1 to 1000, from left to right.
- All episode begin near the center on state 500.
- States 1 and 1000 are terminal states, and it fetches a reward of -1 and 1 on transitioning to them, respectively.
- Policy:
  - Agent can transition from current state to one of 100 states in left or rights, all with equal probability. So there are 200 possible actions. At edges, if an action goes beyond 1 or 1000, then the agent instead transition to the terminal states.

50 tilings

5 buckets (16 buckets)

4 offset

6

1 200 400 600 800 1000

T1

4 203 204

T2 1 - 3, 4 203 204 [ ]$_6$

[ ]$_6$

T50

[ ]$_6$ x 50

50 values = 1

[ ]$_{6 \times 50}$

Neural Networks

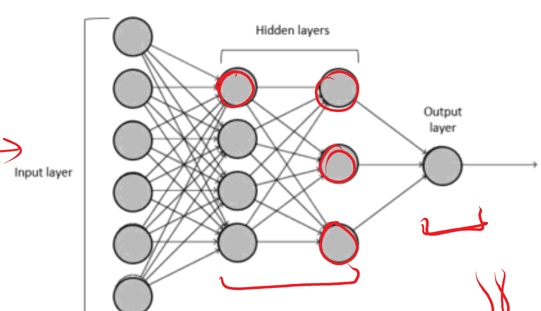
Sunday, May 17, 2020 3:18 PM

Reinforcement Learning

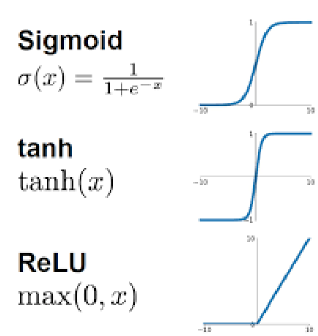
Non-linear function approximator - Artificial Neural Networks

Artificial Neural Networks:

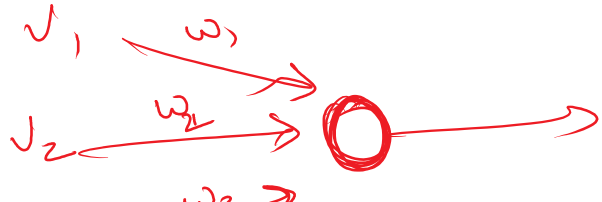
X



$\hat{y}_i$ y_i



ω



$+ v_2 w_2 + v_3 w_3)$

$v_3 \xrightarrow{w_3}$

$$\varphi(v_1 w_1 + v_2 w_2 + v_3 w_3)$$

$$x(s_t) \longrightarrow \boxed{NN(w)} \longrightarrow \hat{v}(s_t, w)$$