南京大学
人工智能学院
School of Artificial Intelligence, Nanjing University

Lecture 6
Value Function Approximation

Tabular presentation, so far

$$Q =$$

s	0	10
	1	12
c	0	8
	1	7
r	0	14
	1	10

We need to store $|S||A|$ number of cells

What if there are too many states ?

note: cells are independent and isolated

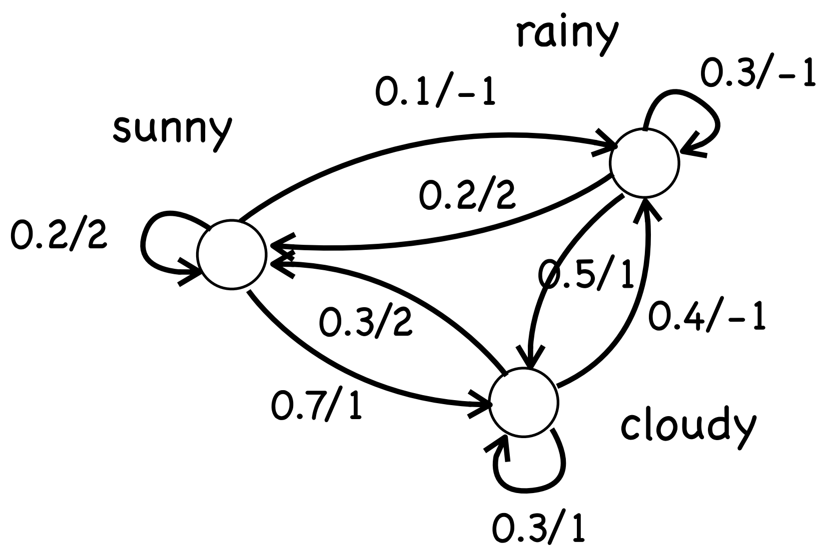
cannot generalize to similar states

What if there are infinite number of states (continuous)?

What if the action is continuous?

Feature vectors

MDP with state ID



State feature vector

[temperature, lightness, humidity, rainfall]

feature vectors also relates the states

e.g. sunny is close to cloudy than rainy

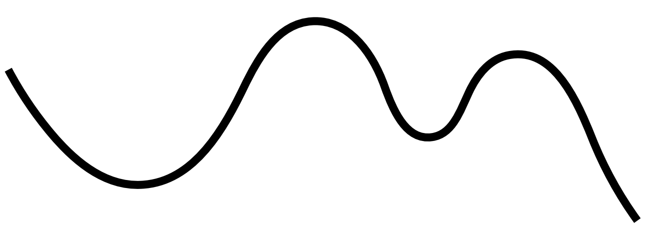
allows generalization over states

Value function approximation

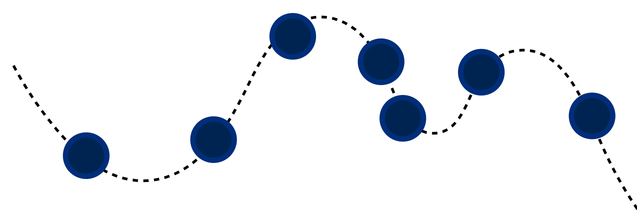
function approximation

as a supervised learning problem

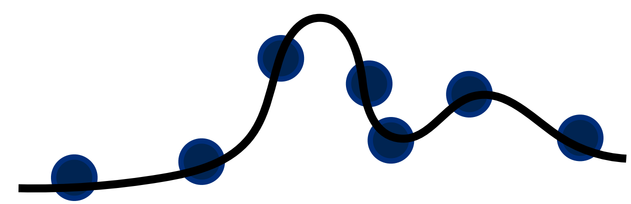
function *f*



data



approximation *h*



Value function approximation

value function approximation

V value $\quad s_t \longrightarrow h(s) \longrightarrow V(s)$

Q value $\quad s_t, a_t \longrightarrow h(s,a) \longrightarrow Q(s, a)$

commonly, we parameterize the approximation function with parameter θ

$$V_\theta(s_t), Q_\theta(s_t, a_t)$$

What are the approximation models?

exactly the supervised learning models

– linear models
– linear models with kernels
– nearest neighbors
– decision trees
– neural networks
–...

RL usually requires more complex models than SL

Approximation objective

learning a model that approximate the true value func.

$$J(w) = E_{s\sim\pi}\Big(V^\pi(s) - V_w(s)\Big)^2$$

$$J(w) = E_{s,a\sim\pi}\Big(Q^\pi(s,a) - Q_w(s,a)\Big)^2$$

why mean square?
V and Q are expectations, mean square leads to unbiased approximation

Let μ^π denote the stationary distribution of states following π

$$J(w) = \int_S \mu^\pi(s)\Big(V^\pi(s) - V_w(s)\Big)^2 \,\mathrm{d}s$$

$$J(w) = \int_S \mu^\pi(s)\int_A \pi(a|s)\Big(Q^\pi(s,a) - Q_w(s,a)\Big)^2 \,\mathrm{d}a\,\mathrm{d}s$$

Solve the parameters

$$w^* = \arg\min J(w) = \arg\min E_{s,a\sim\pi}\Big(Q^\pi(s,a) - Q_w(s,a)\Big)^2$$

for one state-action data sample

online environment: stochastic gradient on single sample

$$w^* = \arg\min \Big(Q^\pi(s,a) - Q_w(s,a)\Big)^2$$

how to solve ? assume differentiable

$$\partial_w J(w) = -2(Q^\pi(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)$$

update *w* towards negative derivative

$$\Delta w = \alpha(Q^\pi(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)$$

Recall the Q update rules

$$\Delta w = \alpha(Q^\pi(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)$$

Recall the errors:

MC update: $Q(s_t,a_t) += \alpha(\underbrace{R}_{\text{target}} - \underbrace{Q(s_t,a_t)}_{\text{model}})$

TD update: $Q(s_t,a_t) += \alpha(\underbrace{r_{t+1} + \gamma Q(s_{t+1},a_{t+1})}_{\text{target}} - \underbrace{Q(s_t,a_t)}_{\text{model}})$

target

model

replace Q table updates by parameter updates

Update with value function approximation

MC update:

$$\Delta w = \alpha(R - Q_w(s_t, a_t))\nabla_w Q_w(s_t, a_t)$$

TD update:

$$\Delta w = \alpha(r_{t+1} + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t))\nabla_w Q_w(s_t, a_t)$$

MC RL with function approximation

$w = 0$

for i=0, 1, ..., m

  generate trajectory <s_0, a_0, r_1, s_1, ..., s_T> by π_ϵ

  for t=0, 1, ..., T-1

    R = sum of rewards from t to $T \times \prod_{i=t+1}^{T-1} \frac{\pi(s_i, a_i)}{p_i}$

    $w = w + \alpha(R - Q_w(s_t, a_t))\nabla_w Q_w(s_t, a_t)$

  end for

  update policy $\pi(s) = \arg\max_a Q_w(s, a)$

end for

Q-learning with function approximation

$w = 0$, initial state

for i=0, 1, ...

$\quad s', r =$ do action from policy π_ϵ

$\quad a' = \pi(s')$

$\quad w = w + \alpha(r + \gamma \boxed{Q_w(s', a')} - \boxed{Q_w(s, a)})\nabla_w Q_w(s, a)$

$\quad \pi(s) = \arg\max_a Q_w(s, a)$

$\quad s = s', a = a'$

end for

Approximation model

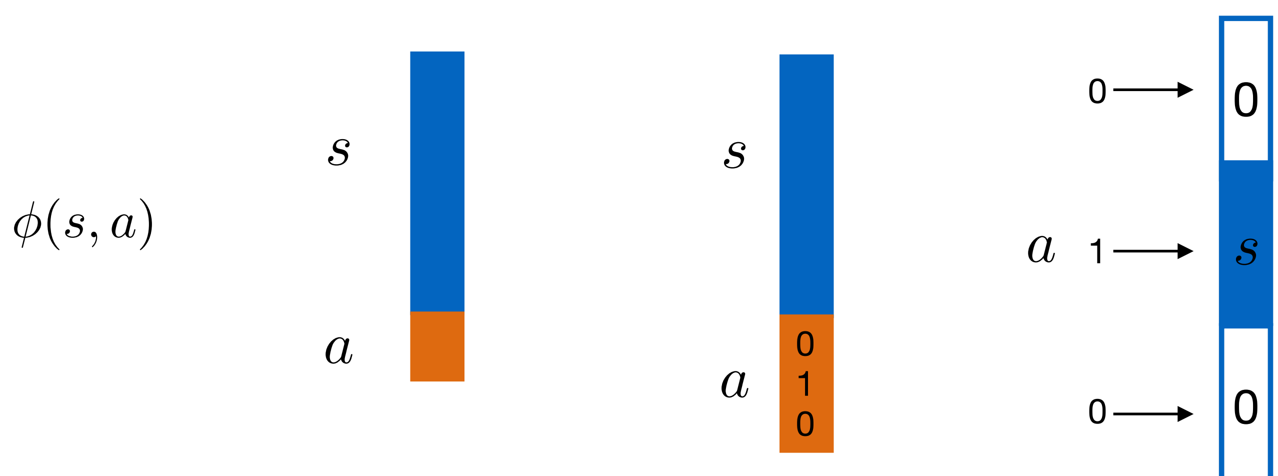
Linear model

encode state-actions into one vector $\phi(s, a)$

$$Q_w(s, a) = w^\top \phi(s, a)$$

$$\nabla_w Q_w(s, a) = \phi(s, a)$$

the encoding is crucial



Approximation model

Linear model

each action has a vector

$$Q_{w_a}(s) = w_a^\top \phi(s)$$

$$\nabla_{w_a} Q_{w_a}(s) = \phi(s)$$

question: can we modify Q(s,a) or the policy easily?

Approximation model

Linear model with kernels

kernel trick brings nonlinearity into linear model

example: $K(x, y|\text{width}) = exp(\frac{-\|x-y\|}{\text{width} \cdot \sigma^2})$

given a set of “training data” $(s_i,\ a_i)$

and a kernel function K

$$Q_w(s,a) = \sum_{i=1}^{m} w_i K(\phi(s,a), \phi(s_i, a_i))$$

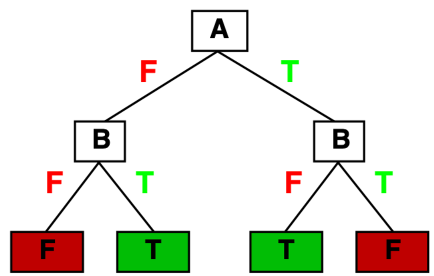
$$\nabla_w Q_w(s,a) = \left[K(\phi(s,a), \phi(s_1,a_1)), K(\phi(s,a), \phi(s_2,a_2)), \ldots K(\phi(s,a), \phi(s_m,a_m))\right]$$

Approximation model

Decision-tree model

decision-tree model can do regression

use regression to learn Q values



decision-tree model is not differentiable

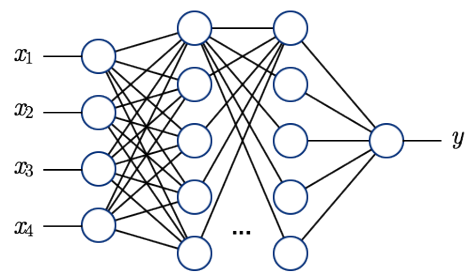
model-update is a re-train

decision-tree model is more interpretable

Approximation model

Neural network model

$\phi(s, a)$



$Q_w(s, a)$

parameters w

Neural network model is differentiable

$$w = w + \alpha(r + \gamma Q_w(s', a') - Q_w(s, a)) \underline{\nabla_w Q_w(s, a)}$$

follow the BP rule
to pass the gradient

Batch update

learning on single sample introduces large variance, particularly for high-capacity models

Batch mode is straightforward:

collect trajectory and history data

$$D = \{(s_1, V_1^\pi), (s_2, V_2^\pi), \ldots, (s_m, V_m^\pi)\}$$

solve batch least square objective

$$J(w) = E_D[\big(V^\pi - \hat{V}(s)\big)^2]$$

linear function: closed form

neural networks: batch update/repeated stochastic update

LSMC, LSTD, LSTD(λ)

Batch methods

Batch mode policy iteration: LSPI with linear model

$Q_0 = 0$, initial state

for i=0, 1, ...

collect data $D = \{(s_1, a_1), (s_2, a_2), \ldots\}$

$$w = \arg\min_w \sum_{(s,a)\in D} (r + \gamma Q_w(s, \pi(s)) - Q_w(s,a))\phi(s,a)$$

$$\pi(s) = \arg\max_a Q_w(s,a)$$

end for

More objectives

MSE: mean square error TD

$$J(w) = E_{s\sim\pi}\Big(V^\pi(s) - V_w(s)\Big)^2$$

MSBE: mean square Bellman error GTD (gradient TD)

$$J(w) = E_{s\sim\pi}\Big(TV_w(s) - V_w(s)\Big)^2$$

$$TV_w(s) = E_{s'\sim P(s,\pi(s))}[R(s,\pi(s),s') + \gamma V_w(s')]$$

[Baird, L. C. Residual algorithms: Reinforcement learning with function approximation. In ICML'95]
[Baird, L. C. Reinforcement Learning Through Gradient Descent. PhD thesis, Carnegie-Mellon University, 1999]
[Hamid Reza Maei. Gradient Temporal-Difference Learning Algorithms. PhD thesis, University of Alberta, 2011. https://era.library.ualberta.ca/items/fd55edcb-ce47-4f84-84e2-be281d27b16a/view/373459a7-72d1-4de2-bcd5-5f51e2f745e9/Hamid_Maei_PhDThesis.pdf]

An example representation should be considered

MSBE: mean square Bellman error

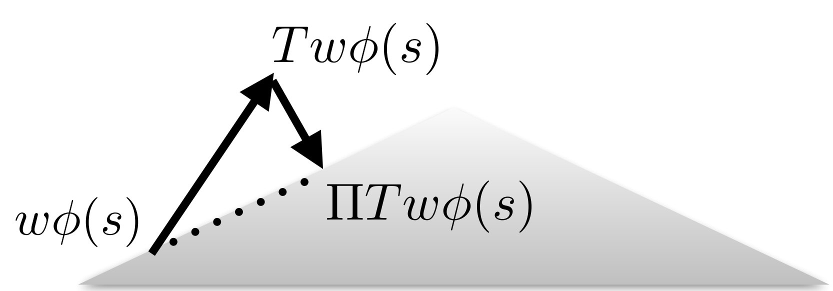
$$J(w) = E_{s\sim\pi}\Big(Tw\phi(s) - w^\top \phi(s)\Big)^2$$

$$Tw\phi(s) = E_{s'\sim P(s,\pi(s))}[R(s,\pi(s),s') + \gamma w^\top \phi(s')]$$

can be out of the representation space of w

MSPBE: mean square projected Bellman error GTD2

$$J(w) = E_{s\sim\pi}\Big(\Pi Tw\phi(s) - w^\top \phi(s)\Big)^2$$



[Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. ICML 2009: 993-1000]