

不像监督学习，不是一次决定，需要一直做决定

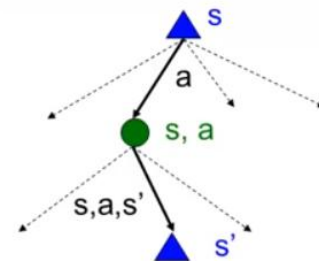
deterministic model : given s 和 a , s' is deterministic not random !!

stochastic model 随机的

Recap: MDPs

▪ Markov decision processes:

- States S
- Actions A
- ▪ Transitions $P(s' | s, a)$ (or $T(s, a, s')$)
- ▪ Rewards $R(s, a, s')$ (and discount γ)
- Start state s_0



▪ Quantities:

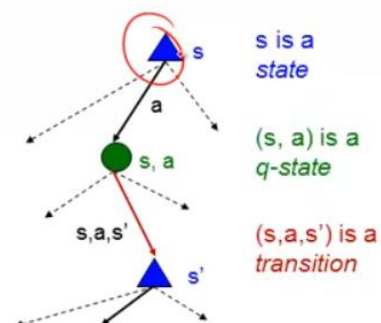
- ▪ Policy = map of states to actions
- ▪ Utility = sum of discounted rewards
- ▪ Values = expected future utility from a state (max node)
- Q-Values = expected future utility from a q-state (chance node)

▪ The value (utility) of a state s :

$V^*(s)$ = expected utility starting in s and acting optimally

▪ The value (utility) of a q-state (s, a) :

$Q^*(s, a)$ = expected utility starting out having taken action a from state s and (thereafter) acting optimally



▪ The optimal policy:

$\pi^*(s)$ = optimal action from state s

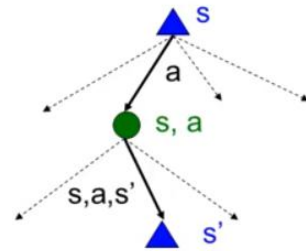
The Bellman Equations

- Definition of “optimal utility” via expectimax recurrence gives a simple one-step lookahead relationship amongst optimal utility values

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



- These are the Bellman equations, and they characterize optimal values in a way we'll use over and over

$V^*(s)$ 给定 state s 的 value function !!!

$Q^*(s, a)$ 给定 state s 和 action a 的 value function

从 s 开始, 已经 take action a 的情况下的 accumulated reward, 不一定是哪个 state

V 和 Q 本质一样!!!! 都是 accumulated reward

只是 Q 的 state 没有确定!!!

$$V^*(s) = \max_a Q^*(s, a)$$

$Q^*(s, a)$: 已经 take action, immediate reward 已经获得, 此选择 $v^*(s')$ 的 s' 时, 就是 $v^*(s)$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

可以 compute each other

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

Value 不同于 Utility, 是期望!!!

MDP

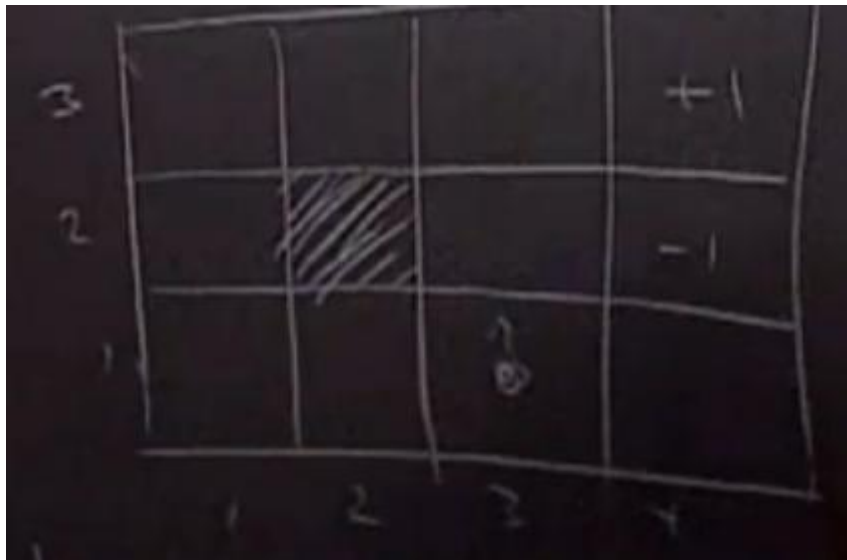
5tuple: (S 状态集 A 行为集 P γ R)

P_{sa} transition distribution, 对于给定 **state**, action 转向下一个 state s' 是个概率分布 $\sum P(s') = 1$

假设一个 grid, 向上概率 0.8, 向左 0.1, 向右 0.1

γ discount factor

R reward function: map state s to Real numbers ($s \rightarrow R$)



$$\begin{aligned}
 P_{(3,1),N}((3,2)) &= 0.8 \\
 P_{(3,1),N}((4,1)) &= 0.1 \\
 P_{(3,1),N}((2,1)) &= 0.1 \\
 P_{(3,1),N}((3,3)) &= 0
 \end{aligned}$$

$P_{(s)N}(s')$ transition function

从 s 出发，到达 s' 的概率， s' 是随机变量!!!

Reward function

$$\begin{aligned}
 R((4,3)) &= +1 \\
 R((4,2)) &= -1 \\
 R(s) &= -0.02 \\
 &\text{for all other states}
 \end{aligned}$$

at state s_0

choose a_0

get to $s_1 \sim P_{s_0 a_0}$ (draw random from $P_{s_0 a_0}$)

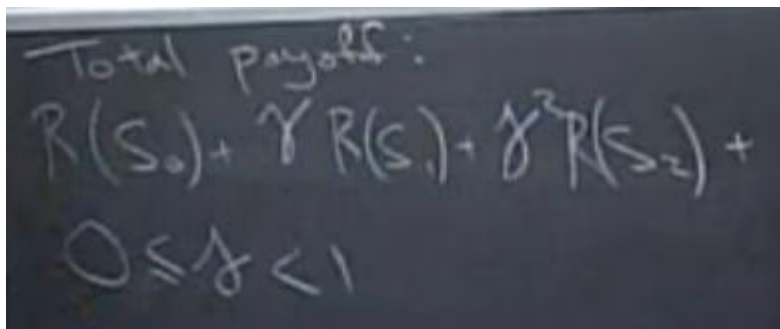
chooser a_1

get to $s_2 \sim P_{s_1 a_1}$

直到结束

得到一个 state sequence, how well the sequence

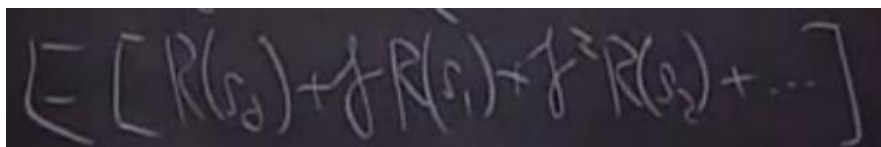
total payoff 薪酬 也被成为 gain



Handwritten formula for Total payoff: $R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$ with $0 \leq \gamma < 1$

金钱在贬值, 未来的钱越来越不值钱

Value function: $V(s)$ expection of total payoff of s



Handwritten formula for Value function: $V(s) = [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$

当 $s_0 = s$

$V(s)$ 是期望, $s_1, s_2 \dots$ 是从 s 开始到下个 state, 是随机变量,
要求他们的期望, 希望 $V(s)$ 最大

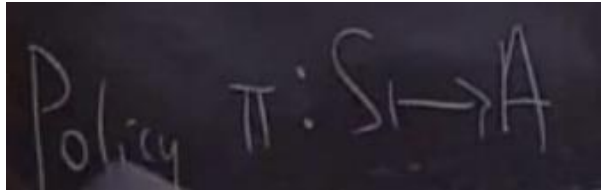
goal:

choose action(a_1, a_2, \dots)

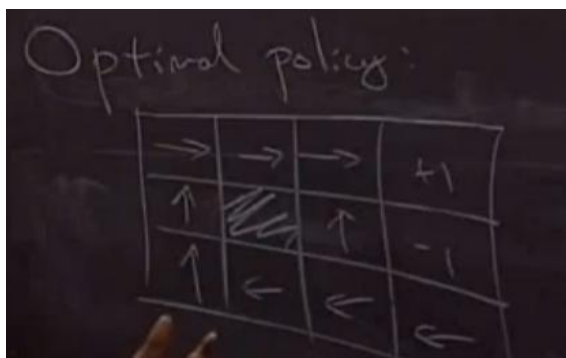
to maximize the s **expection of total payoff of s**

也就是用强化学习算法，计算一个 **optimal policy (denote by π^*)**

Policy π is function map one state to one action



得到这个 policy 函数我们就可以，知道每一 state 应该如何走



上面是一个 **optimal policy**，每个 state 应该如何 action

π^* **optimal policy maximize expect total value payoff**

给定一个 π 就是一个 policy

V^π

For any π , define a value function $V^\pi: S \Rightarrow R$

s.t $V^\pi(s)$ is expectation of total payoff if **start in state s, and execute π** 。也就是从 s 开始按照 **policy** 走到 **end**，而最终得到的 **total payoff**，是 long term reward

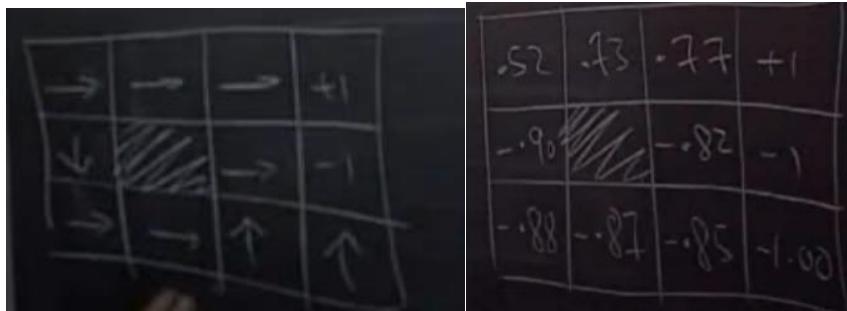
给定一个 π ，所有 state 的 $V^\pi(s)$ 就固定了!!! 见下图

期望是因为 $s_1 s_2 \dots$ 都是随机变量

$$V^\pi(s) = E(R(s_0) + \gamma R(s_1) + \dots | s_0 = s, \pi)$$

例子：

given a policy π : 得到右边的 V^π value function:



可以看出右下角的 $V^\pi(s)$ 很小，因为从这个 state 开始，采取 π ，就到了 -1 而结束，说明是 bad π !!!

变形下：

$$V^\pi(s) = E(R(s_0) + \gamma(R(s_1) + \gamma R(s_2) + \dots) | s_0 = s, \pi)$$

$R(s_0)$ immediate reward 后面是 feature reward

括号里面 $(R(s_1) + \gamma R(s_2) + \dots)$ 是 $V^\pi(s_1)$

因此转换成了，只需考虑一步!!!

bellman equation 核心思想：

因为 $V^\pi(s)$ 是从 s 起 total payoff，而 $R(s)$ 已经固定，只需要计算下一步 state 的 $V^\pi(s')$ ，就可以计算 $V^\pi(s)$

$$V^\pi(s) = E(R(s) + \gamma V^\pi(s') | s_0 = s, \pi)$$

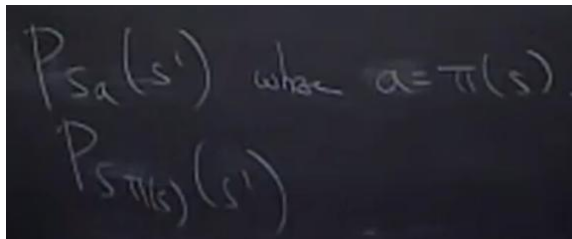
$$V^\pi(s) = R(s) + E(\gamma V^\pi(s')) | s_0 = s, \pi$$

s' 是随机变量，要取期望

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s') \quad \text{bellman equation!!!}$$

$P_{s\pi(s)}(s')$ 是给定起点 s , 和 $\pi(s)$ action, 所对应的下一步 s' (随机变量) 的概率分布

$P_{s\pi(s)}(s')$ 等价 $P_{sa}(s')$:



$P_{sa}(s') \text{ when } a = \pi(s).$
 $P_{s\pi(s)}(s')$

bellman equation help us solve the value func for policy in closed form 封闭形式的一种解法，可以方便解出所有 state 的 $V^\pi(s)$

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s')$$

上面 sum 是线性的!!!

通过解决上面 linear system 线性方程组们就可以解出每个 state 的 $V^\pi(s)$

例子:

起点(3,1) $\pi(3,1)=\text{North}$, 则 value function $V^\pi(3,1)$

The image shows a chalkboard with handwritten mathematical expressions and a diagram. At the top right, it says $\pi((3,1)) = 1$. Below this, the Bellman optimality equation is written for state (3,1): $V^*((3,1)) = R((3,1)) + \gamma \max_a [0.8 V^*((3,2)) + 0.1 V^*((4,1)) + 0.1 V^*((2,1))]$. To the right of the equation is a 4x4 grid representing a state space. The columns are labeled 1, 2, 3, 4 at the bottom. The grid contains some markings, including a circle in the third row, third column and a cross in the second row, second column.

我们目标是解出每个 state 的 value function!!!

我们共有 11 个这样的方程，11 个未知数，可以解出！

我们最终想得到一个让所有 state value 都最大的 π^* ，也就是每一步每个 action 都达到 $V^*(s)$ 。也就是让 π 的每一 action 都达到最优!!!

Optimal value function

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

起点是 s ，遍历所有的策略 π ，取 $V(s)$ 最大的值

π^* 让 $V(s)$ 取最大值

$$V^*(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | \pi^*, s_0 = s]$$

$$V^*(s) = R(s_0) + \gamma E[(R(s_1) + \gamma R(s_2) + \dots) | \pi^*, s_0 = s]$$

以上可以看出一个重要规律，也是最难点：

$R(s_0)$ 是常数，给定最佳 π^* 下， s 取 **optimal value**，就要求 s' 也要取 **optimal value**， $V^*(s)$ 把任务交给 $V^*(s')$ ，即

$$V^*(s) = R(s_0) + \gamma V^*(s') \quad \text{递归式}$$

也表示在最佳 π^* 下，所有 **state** 都取 **optimal value**

存在这样一个策略，使得所有 **state** 都取 **optimal value**

$$\text{又 } V^*(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | \pi^*, s_0 = s]$$

$$\text{即 } V^*(s) = \max_{\pi} E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s]$$

$$V^*(s) = R(s) + \max_{\pi} E[\gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

最终得出：**bellman equation**

$$V^*(s) = R(s) + \max_a \gamma \sum P_{sa}(s') V^*(s')$$

$P_{sa}(s')$ 只是 s transit to s' 的概率分布

mean: 当前 s 下采取 a ，使得 $V^*(s')$ 期望最大

$V^*(s')$ 是因为， s' 采取的也是 π^* 最佳策略!!!

所有下一个 **state** 都是 $V^*(s')$ **optimal value**(难点)

bellman equation

immediate reward + 下一个 state value

bellman equation 做的事就算分解，只分解到下一步，得出当前 **state** 和下一个 **state'**(expect future state) 的关系!!! 也就是当前的值，只跟下一个状态值有关，也就是将任务推给

了下一个状态!!!

上面的式子是个方程组， a 是参数，我们要求得每一个 state 的 optimal a (best action)，其实就是求最佳策略函数 π

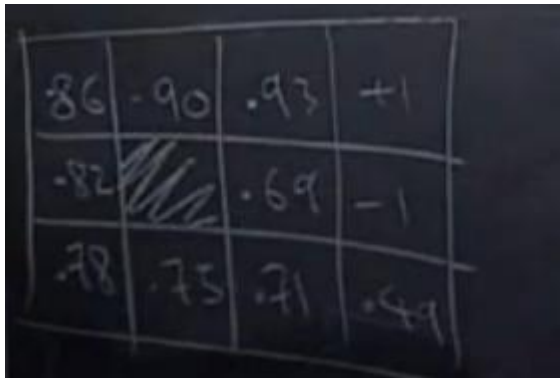
$$\pi^*(s) = \arg \max_a \sum P_{sa}(s') V^*(s') \quad \text{就得到了每个 } s \text{ 新策略函数!}$$

得到最佳策略函数，式子是上面化简后的结果

解释：我们处于当前 state s ，我们想 choose 一个最佳 action 最大化 **expect future state** value!!

例子：

$$V^*(s)$$



0.86	0.90	0.93	+1
0.82	0.82	0.69	-1
0.78	0.75	0.71	0.49

加入在(3,1)state

$a=\text{west}$ 时 value 值,向左 0.8 概率向上 0.1，不动 0.1

maximize the value

$$\begin{aligned}
 W &:= \sum_{s'} P_{sa}(s') V^*(s') \\
 &= 0.8 \times 7.5 + 0.1 \times 6.9 \\
 &\quad + 0.1 \times 7.1 = 0.76
 \end{aligned}$$

但 policy π 太多了，不可能 exhaust 穷尽

方法：

Value iteration algorithm

$$\begin{aligned}
 &\text{Initialize } V(s) = 0 \quad \forall s. \\
 &\text{For every } s, \text{ update} \\
 &\quad V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')
 \end{aligned}$$

1. For each state s , initialize $V(s) := 0$.
2. Repeat until convergence {

For every state, update $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s') V(s')$.

第二步类似于后向传播

R, P, V, A 都已知, 每个 state, 穷尽 A , 选择最大的 $V(s')$

update 策略:

1 synchronous update :simutaneously 计算右边式子同时更新所有状态

2 asynchronous update update a state one time, 右面的值会一直变, 随着更新后

最后 may converge: $V(s) \rightarrow V^*(s)$

计算出 $V^*(s)$

算法证明比较复杂!!!

policy iteration

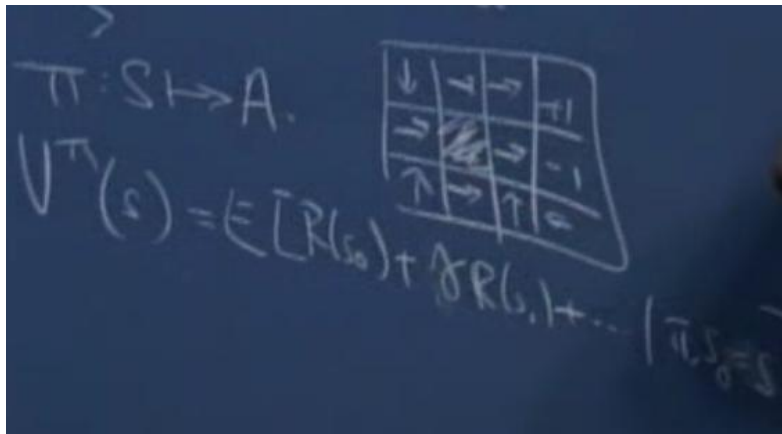
Apart from value iteration, there is a second standard algorithm for finding an optimal policy for an MDP. The **policy iteration algorithm**

1. Initialize π randomly.
2. Repeat until convergence {
 - (a) Let $V := V^\pi$.
 - (b) For each state s , let $\pi(s) := \arg \max_{a \in A} \sum_{s'} P_{sa}(s') V(s')$.}

given random initial policy π

repeat{

1 given fixed π , to solve for V^π (if you execute π , from that state, the value you got)



例如计算 s_0 的 V^π

2 update 每个 state, 穷尽 A, 选择最大的 $V(s')$

$$\pi^*(s) := \arg \max_a \sum P_{sa}(s') V^*(s')$$

得到新的 policy 函数, 再进行迭代

}

最后 may converge: $V(s) \rightarrow V^*(s) \quad \pi(s) \rightarrow \pi^*(s)$

trade off: 上面算法最计算消耗的是 $V := V^\pi$

需要用 bellman equation 方程组, 解出每个 $V^\pi(s)$, 再赋给每个 V , 每次 iterate 就要解一次方程组

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s')$$

如果 state 太多, 选择 value iteration 算法

到目前为止, given a 5tuple: (S 状态集 A 行为集 P γ R)

我们可以得到一个最优的 policy!!!

但是 P_{sa} transition probability 我们往往不清楚
需要 learn from data to estimate

$$P_{sa}(s') = \frac{\text{从s, take action a, 到s'次数}}{\text{从s take action a的次数}}$$

如果分子或分母为 0, $P_{sa}(s') = \frac{1}{|S|}$

put together: estimate P and 求得 policy

repeat: {

1 take action using some π to get experience for some number of trials. (execute policy π , observe state transition)

2 update estimate of $P_{sa}(s')$, base on 上面 experience

3 value iteration 尝试所有 A, 得到 max V

solve bellman equation of V^* get estimate V

(Apply value iteration with the estimated state transition probabilities and rewards to get a new estimated value function V)

(question 如果 initialize use the value of previous round ,
converge faster!!!)

4 用上面得到的各个 state 的 action 更新

$$\text{update } \pi^*(s) := \arg \max_a \sum P_{sa}(s') V^*(s')$$

(Update π to be the greedy policy with respect to V)

}

1. Initialize π randomly.
2. Repeat {
 - (a) Execute π in the MDP for some number of trials.
 - (b) Using the accumulated experience in the MDP, update our estimates for P_{sa} (and R , if applicable).
 - (c) Apply value iteration with the estimated state transition probabilities and rewards to get a new estimated value function V .
 - (d) Update π to be the greedy policy with respect to V .}

MDP 总结，太精彩了!!!!

最终 state 叫 absorbing state

recap

MDP: $(S, A, \{P_{sa}\}, \gamma, R)$

11 states.

$A = \{N, S, E, W\}$

P_{sa} at absorbing state

$R = \begin{cases} +1 \\ -1 \end{cases}$ at absorbing state

$\gamma = 0.99$ elsewhere.

$\pi: S \rightarrow A$

$V^\pi(s) = E[R(s_0) + \gamma R(s_1) + \dots | \pi, s_0 = s]$

Find $V^*(s) = \max_{\pi} V^\pi(s)$

ST

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V^*(s')$$

$$VI: V(s) := R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V(s')$$

VI(value iteration) compute v^*

once we find the v^* , we can compute π^*

E.g.

.86	.90	.92	1
.82	.85	.69	1
.79	.75	.71	.49

V^*

$$\pi^*(s) = \arg \max_a \sum_{s'} P_{sa}(s') V^*(s')$$

continuous state

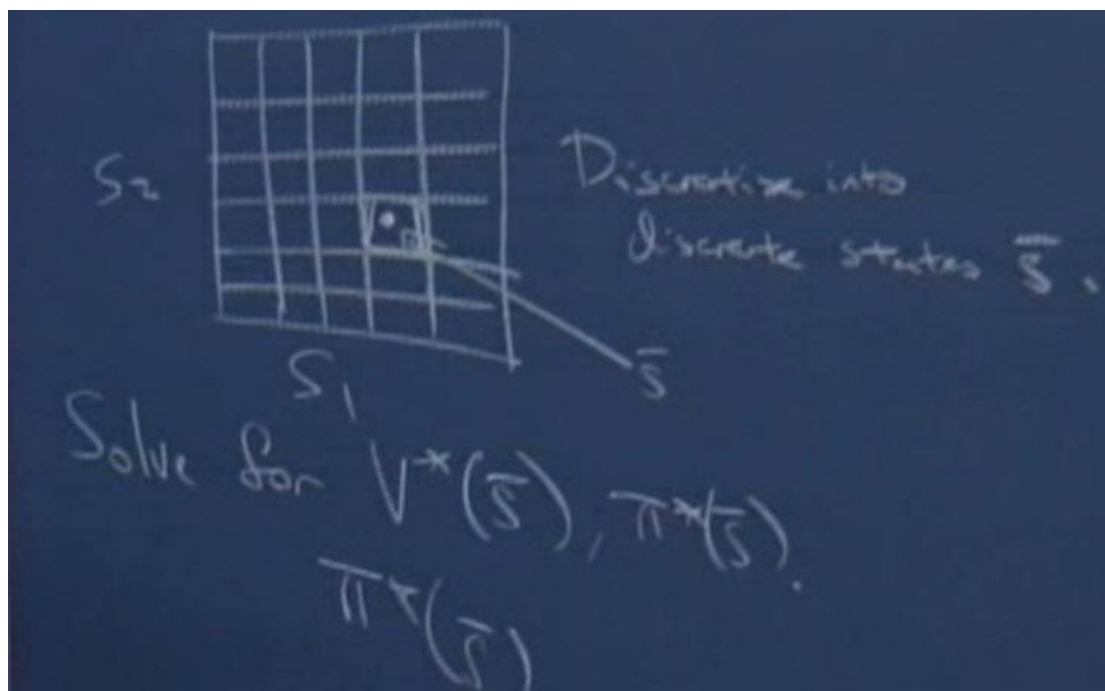
state 可能有很多维度

如 car: (x,y, orientation,v) 4 个维度 而且是连续的

discretization

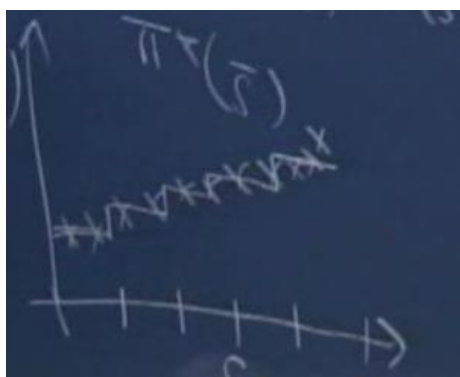
transfer continuous state problem with a finite or discrete set of states and then you can use policy iteration or value iteration to solve for $V^*(s)$ -bar

假设有 **2dimension continuous state variables**



将连续的值，划分为不同的范围 interval state, 处于某个值，就知道属于某个 state

piecewise function 分段函数



approximate this function using something that's piecewise constant

1 这种代表性不好

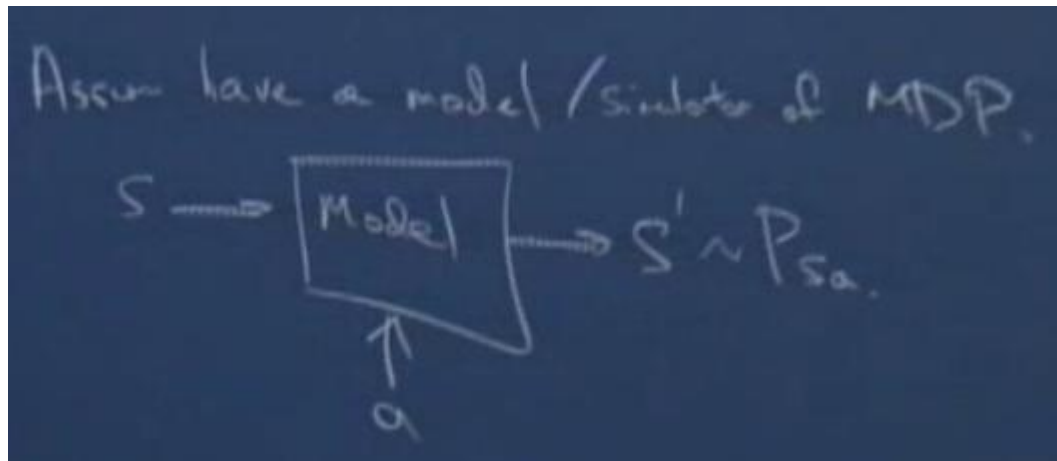
2 数据维度太大，每个 state 就是一个变量

对于维度小的，如上面 2dimension

Value function approximation

我们另外的更好方法 **approximate V^* directly**

解决问题: **Continuous State, discrete Action**



每次 input, 相当于一个随机试验, s' 是随机变量, 随机变量分别服从 P_{sa} , 如何确定 model?

方法: learn a model

参看讲义: $\phi(s)$ 是 s feature 的一个 map, feature 的一个组合, 和 create new feature.

Handwritten notes on a dark blue background. The text reads: "Choose feature $\phi(s)$ of state s . Could $\phi(s) = s$. E.g., $\phi(s) = \begin{bmatrix} 1 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ 0.0 x_1 x_2 \\ 0.1 x \end{bmatrix}$ ".

$$V^* \mapsto \pi^*$$

$$\pi^*(s) = \arg \max_a \mathbb{E}_{s' \sim p_{sa}} [V^*(s')] \\ s = (x, \theta, \dot{x}, \dot{\theta})$$

$$a = \arg \max_a \mathbb{E}_{s' \sim p_{sa}} [V^*(s')]$$

得到 V^* 如何计算 π^* 对于 continuous state

只计算当 robot 或 system 的 some specific state 的 action, 如上

The optimal action

$$V(f(s, a)) = \theta^T \phi(s')$$

$$\text{where, } s' = f(s, a)$$

词汇：暂时的 temporarily 放下 put aside chop 砍

obstacle 障碍 关联 associate common practice 惯例

helicopter 直升机 resort to 求助于 度假胜地

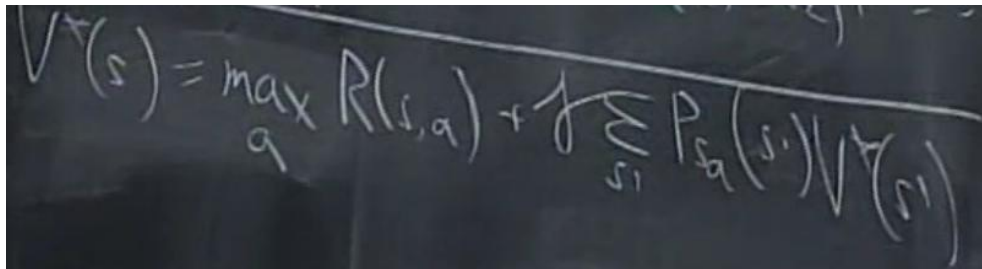
More general model

change Reward function

State-action Reward : $R : S \times A \Rightarrow R$

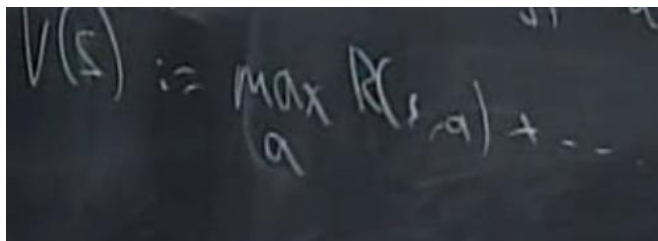
total payoff; $R(s_0, a_0) + \gamma R(s_1, a_1) + \dots$

可以帮助我们 model problem in which different actions have different cost(机器人, 移动 or 不动, 耗油不同, cost 不同)
reward 不仅跟状态有关, 还和 action 有关!


$$V^+(s) = \max_a R(s, a) + \gamma \sum_{s'} P_{sa}(s') V^+(s')$$

因为 immediate reward 也包含参数 a, 因此 max 要提在外面

VI:


$$V(s) := \max_a R(s, a) + \dots$$

$$\pi^*(s) = \arg \max_a R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s')$$

通过 V^* , 就得到 π^*

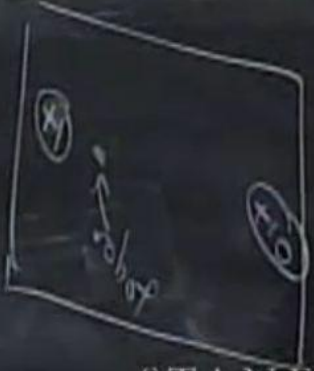
Finite horizon MDPS, 一般不用加 γ

也就是时间有限最大是 T , 每个 state, reward 对应不同时间:

不同时间 cost 不同, 因此 reward 也不同 $R^{(t)}(s_0, a_0)$

Finite horizon MDPs: $(S, A, \{P_{sa}^{(t)}\}, T, R)$
 Payoff: $T = \text{horizon time}$
 $R(s_0, a_0) + R(s_1, a_1) + \dots + R(s_T, a_T)$
 Optimal policy: non-stationary.

$s_{t+1} \sim P^{(t)}(s_{t+1} | s_t, a_t)$



STANFORD UNIVERSITY

最佳策略, non-stationary

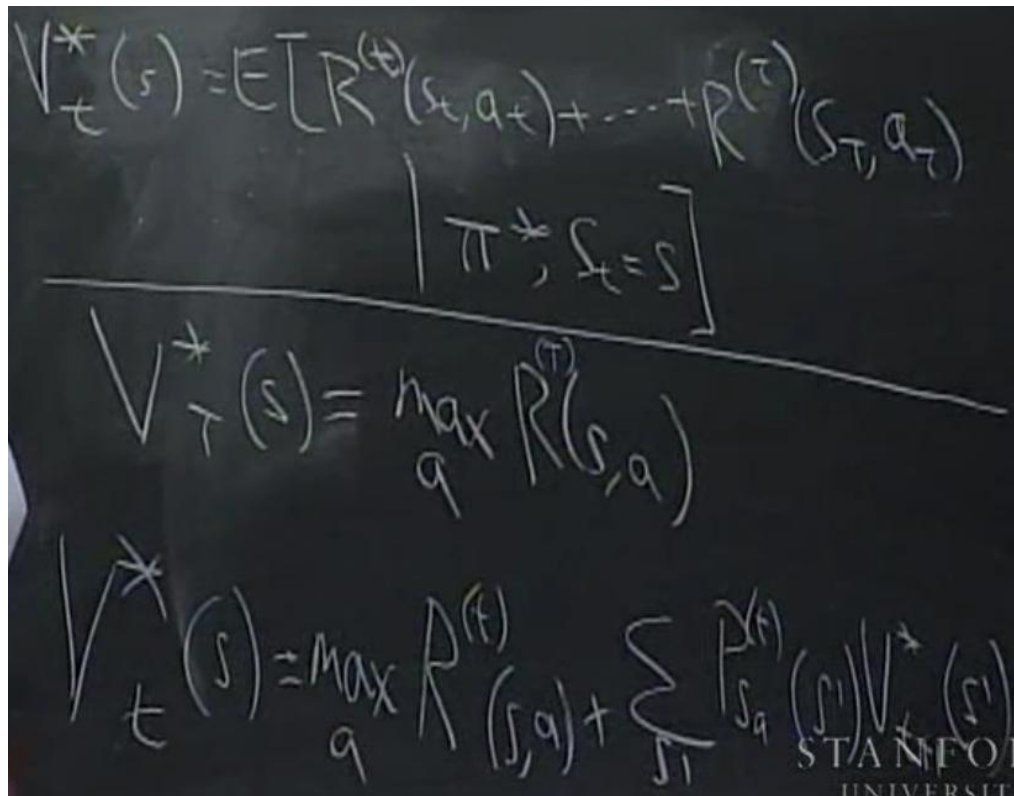
有图, 左边+1, 右边=10, 但由于 finite T , 只能取左边近的!

所有 state transition probability $P_{sa}^{(t)}$ 也随着时间变化, 是

non-stationary

如何找到 optimal policy?

先求 $V^*(s)$


$$V_t^*(s) = E[R^{(t)}(s, a_t) + \dots + R^{(T)}(s_T, a_T) \mid \pi^*, s_t = s]$$

$$V_T^*(s) = \max_a R^{(T)}(s, a)$$
$$V_t^*(s) = \max_a R^{(t)}(s, a) + \sum_{s'} P_{sa}^{(t)}(s') V_{t+1}^*(s')$$

1 $V_t^*(s)$ 是在 s 下, t 时刻开始时的 optimal value

等于在采取最佳策略 π^* 的情况下得到的 total payoff 期望!!!

2 转成 bellman equation:

$$V_t^*(s) = \max_a R^{(t)}(s, a) + \sum_{s'} P_{sa}^{(t)}(s') V_{t+1}^*(s')$$

以上是递归是, 递归的开始: 最后一个时刻 T (最后一次 action)

$$V_T^*(s) = \max_a R^{(T)}(s, a)$$

use dynamic programming algorithm

从后往前计算 $V_{T-1}^*(s)$, $V_{T-2}^*(s)$

$$\pi_t^*(s) = \arg \max_a R(s,a) + \sum_{s'} P_a(s') V_{t+1}^*(s')$$

依次计算: $\pi_T^*(s), \pi_{T-1}^*(s), \pi_{T-2}^*(s)$

以上是用 use dynamic programming algorithm for finite MDPS

Linear Quadratic Regulation(LQR)

non-stationary dynamic A,B always change

Linear Quadratic Regulation (LQR).

$S, A, \{P_{s_a}\}, T, R.$

$S = \mathbb{R}^n. \quad A = \mathbb{R}^d.$

$P_{s_a}: \quad S_{t+1} = A_t S_t + B_t a_t + w_t$

$w_t \sim \mathcal{N}(0, \Sigma_w)$

$A_t \in \mathbb{R}^{n \times n}$

$B_t \in \mathbb{R}^{n \times d}$

STANFORD UNIVERSITY

P 还是用 regression A,B 的维度? question

w N (均值 0, 协方差矩阵)

Quadratic Reward function:

$$R(s_t, a_t) = -(s_t^T U_t s_t + a_t^T V_t a_t)$$
$$U_t \in \mathbb{R}^{n \times n}, V_t \in \mathbb{R}^{d \times d}$$
$$U_t \geq 0, V_t \geq 0 \text{ (psd)}$$

U V 都是 Positive Semi-Definite 半正定矩阵

imply:

$$s_t^T U_t s_t \geq 0, a_t^T V_t a_t \geq 0$$
$$\Rightarrow R(s_t, a_t) \leq 0$$

example: 直升机

Helicopter: Want $s_t \approx 0$.

Choose $U_t = I, V_t = I$.

$$R(s_t, a_t) = -s_t^T s_t - a_t^T a_t$$
$$= -\|s_t\|^2 - \|a_t\|^2$$

求 MDP Model

The image shows handwritten notes on a chalkboard. At the top, the state transition equation is written as $S_{t+1} = A s_t + B a_t$. Below this, two sequences of states and actions are shown. The first sequence is $S_0^{(1)} \xrightarrow{a_1^{(1)}} S_1^{(1)} \xrightarrow{a_2^{(1)}} \dots \rightarrow S_T^{(1)}$. The second sequence is $S_0^{(m)} \xrightarrow{a_1^{(m)}} S_1^{(m)} \xrightarrow{a_2^{(m)}} \dots \rightarrow S_T^{(m)}$. At the bottom, the cost function is written as
$$\arg \min_{A, B} \sum_{i=1}^m \sum_{t=1}^T \|S_{t+1} - (A s_t + B a_t)\|^2$$

try different your system, and watch what state we get to

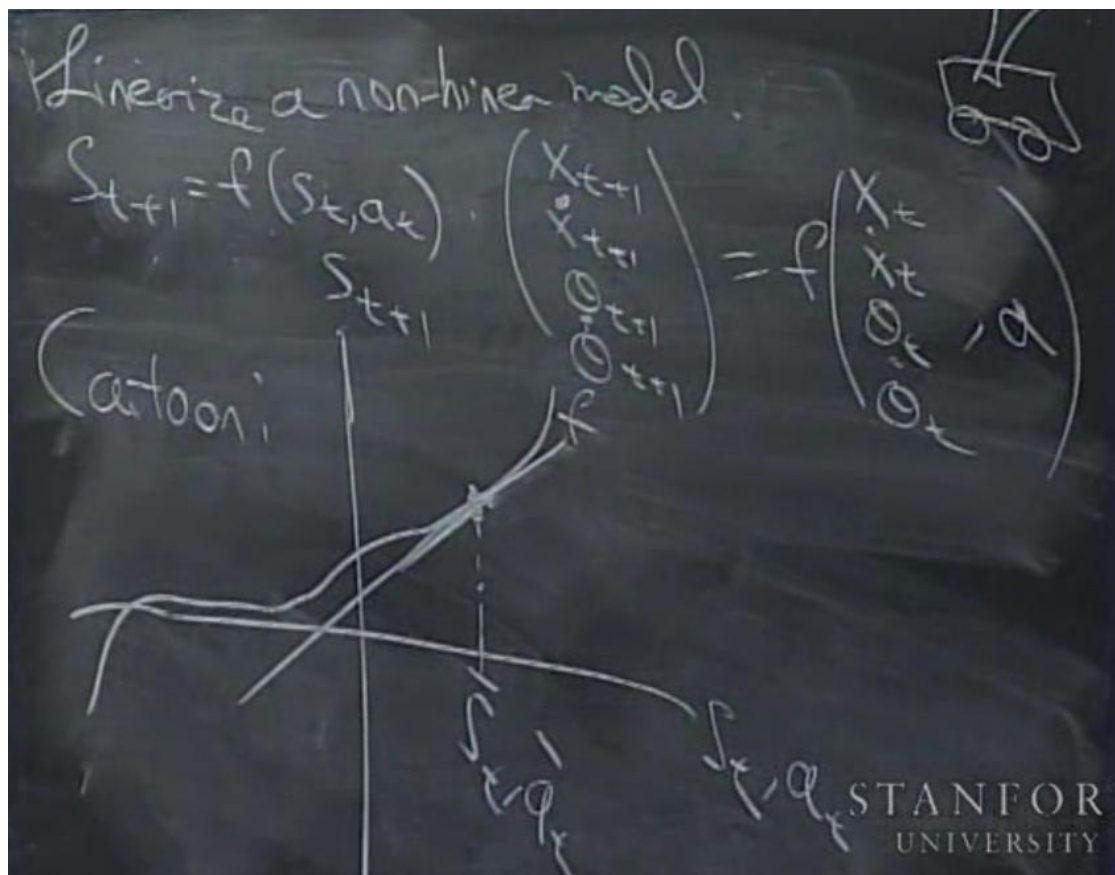
try m time

使得误差平方和最小

另一种求法

linearize a non-linear model

f 是 non linear model



f 函数图如上

—
有一点 \bar{s} ，取这一点的切线，就变成 linear 了！

$$S_{t+1} = f(s_t)$$

$$\underline{S_{t+1}} \approx f'(\bar{s}_t)(\underset{\uparrow}{s_t - \bar{s}_t}) + f(\bar{s}_t)$$

$$\bar{s}_t = \text{constant}$$

点斜式 直线方程(s 是单变量，暂时忽略 a),

—
 \bar{s} 是个常数，上面成了 s_t 和 s_{t+1} 的线性关系式！

当我们 linearize 曲线时，有些地方 approximate 比较好，有些不好，所以要将 \bar{s} 周围的点也作为参数，来拟合直线



choose the position to linearize, where spent most of time

对于 s , a , linearize 一个平面，倒三角是偏导数

$$\begin{aligned} \hat{s}_{t+1} \approx & f(\bar{s}_t, \bar{a}_t) + \left(\nabla_s f(\bar{s}_t, \bar{a}_t) \right)^T \cdot (\hat{s}_t - \bar{s}_t) \\ & + \left(\nabla_a f(\bar{s}_t, \bar{a}_t) \right)^T (\hat{a}_t - \bar{a}_t) \end{aligned}$$

得到了 $s_{t+1} = As_t + Ba_t$

find policy maximize finite horizon reward

$$\max: R^{(1)}(s_0, a_0) + R^{(1)}(s_1, a_1) + \dots + R^{(T)}(s_T, a_T)$$

先找: $V_T^*(s_T) = \max_{a_T} R^{(T)}(s_T, a_T)$

然后再 backforward to $V_{T-1}^*(s_{T-1}) \dots$

$$\begin{aligned} V_T^*(s_T) &= \max_{a_T} R^{(T)}(s_T, a_T) \\ &= \max_{a_T} -s_T^T U_T s_T - a_T^T U_T a_T \end{aligned}$$

上边的 T donte 矩阵 transpose 转置

下边 T time

$$\begin{aligned}
 V_T^*(s_T) &= \max_{a_T} R^{(T)}(s_T, a_T) \\
 &= \max_{a_T} -s_T^T U_T s_T - a_T^T V_T a_T \\
 &= -s_T^T U_T s_T \quad (\text{because } a_T^T V_T a_T \leq 0) \\
 \pi_T^*(s_T) &= \arg \max_{a_T} R^{(T)}(s_T, a_T) = 0
 \end{aligned}$$

question: 因为 UV 都是正定矩阵, 因此乘积都是正的, 整个式子是负的, $\max=0$

右边的为正, 为啥就等于左边的呢???

our goal: dynamic program step: $V_{t+1}^* \rightarrow V_t^*$

$$\begin{aligned}
 V_{t+1}^* &\leadsto V_t^* \\
 V_t^*(s_t) &= \max_{a_t} R^{(t)}(s_t, a_t) + \sum_{s_{t+1}} P_{s_{t+1}}(s_t) V_{t+1}^*(s_{t+1}) \\
 &= \max_{a_t} R^{(t)}(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim P_{s_{t+1}}(s_t, a_t)} [V_{t+1}^*(s_{t+1})]
 \end{aligned}$$

LQR 有个 property, 每个 V_t^* can be represented as quadratic function

Suppose $V_{t+1}^*(s_{t+1}) = s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}$
 $\Phi_{t+1} \in \mathbb{R}^{n \times n}; \psi_{t+1} \in \mathbb{R}$
Can show
 $V_t^*(s_t) = s_t^T \Phi_t s_t + \psi_t$
for some Φ_t, ψ_t .

假设 V_{t+1}^* 是以上 quadratic 形式 Φ 是矩阵, Ψ 是实数
我们带入上面的式子, 可以得到相同的形式 V_t^* , 条件是找到合适的 Φ, Ψ 。LQR 的 property!!!

s 是向量 $(x_1, x_2, x_3)^T$

Φ 是一个 3×3 矩阵

$s^T \Phi s$ 很奇妙, 他是 x_1, x_2, x_3 的所有二次的组合, 依据 Φ 的变化而变化

如:

$$\begin{array}{ccc}
 & & \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \\
 \begin{matrix} x1 & x2 & x3 \end{matrix} & & \begin{matrix} x1 \\ x2 = x1^2 + x2^2 + x3^2 \\ x3 \end{matrix}
 \end{array}$$

start the recursion

$$\text{已知 } V_t^*(s_t) = -s_t^T U_t s_t$$

$$V_t^*(s_t) = s_t^T \Phi_t s_t + \Psi_t$$

从头开始推：

$$V_T^*(s_T) = -s_T^T U_T s_T$$

$$\text{因此, } \Phi_T = -U_T \quad \Psi_T = 0$$

$$V_T^*(s_T) = s_T^T \Phi_T s_T + \Psi_T$$

转为 bellman equation 求解：

$$V_t^*(s_t) = \max_{a_t} [-s_t^T U_t s_t - a_t^T V_t a_t + E_{s_{t+1} \sim N(A_t s_t + B_t a_t, \Sigma_w)} [s_{t+1}^T \Phi_{t+1} s_{t+1} + \Psi_{t+1}]]$$

$\underbrace{-s_t^T U_t s_t - a_t^T V_t a_t}_{P_{s_t a_t}} + E_{s_{t+1} \sim N(A_t s_t + B_t a_t, \Sigma_w)} \underbrace{[s_{t+1}^T \Phi_{t+1} s_{t+1} + \Psi_{t+1}]}_{V_{t+1}^*(s_{t+1})}$

$$V_t^*(s_t) = \max_{a_t} [-s_t^T U_t s_t - a_t^T V_t a_t + E_{s_{t+1} \sim N(A_t s_t + B_t s_t, \varepsilon_w)} (s_{t+1}^T \Phi_{t+1} s_{t+1} + \Psi_{t+1})]$$

$-s_t^T U_t s_t - a_t^T V_t a_t$ 是 immediate reward

$s_{t+1} \sim N(A_t s_t + B_t s_t, \varepsilon_w)$ 下一个 state draw from distribution

因为 $s_{t+1} = A_t s_t + B_t s_t + \varepsilon_w$ 正态分布

$N(A_t s_t + B_t s_t, \varepsilon_w) = P_{s_t a_t}$ state transition probability

加上 $V_{t+1}^*(s_{t+1})$ 的期望

bellman 就是 dynamic programming !!!

展开 expand $E_{s_{t+1} \sim N(A_t s_t + B_t s_t, \varepsilon_w)} (s_{t+1}^T \Phi_{t+1} s_{t+1} + \Psi_{t+1})$ 化简为:

是一个 **quadratic function of** a_t

再回到前面公式：

$$V_t^*(s_t) = \max_{a_t} -s_t^T U_t s_t - a_t^T V_t a_t + E_{s_{t+1} | N(A_t s_t + B_t a_t, \Sigma_w)} [s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}]$$

Simplifies to quadratic function of s_t .

STANFORD UNIVERSITY

只需对 a_t ，求导数令 0，解出 a_t

$$a_t = (B_t^T \Phi_{t+1} B_t - V_t)^{-1} B_t^T \Phi_{t+1} A_t \cdot s_t$$

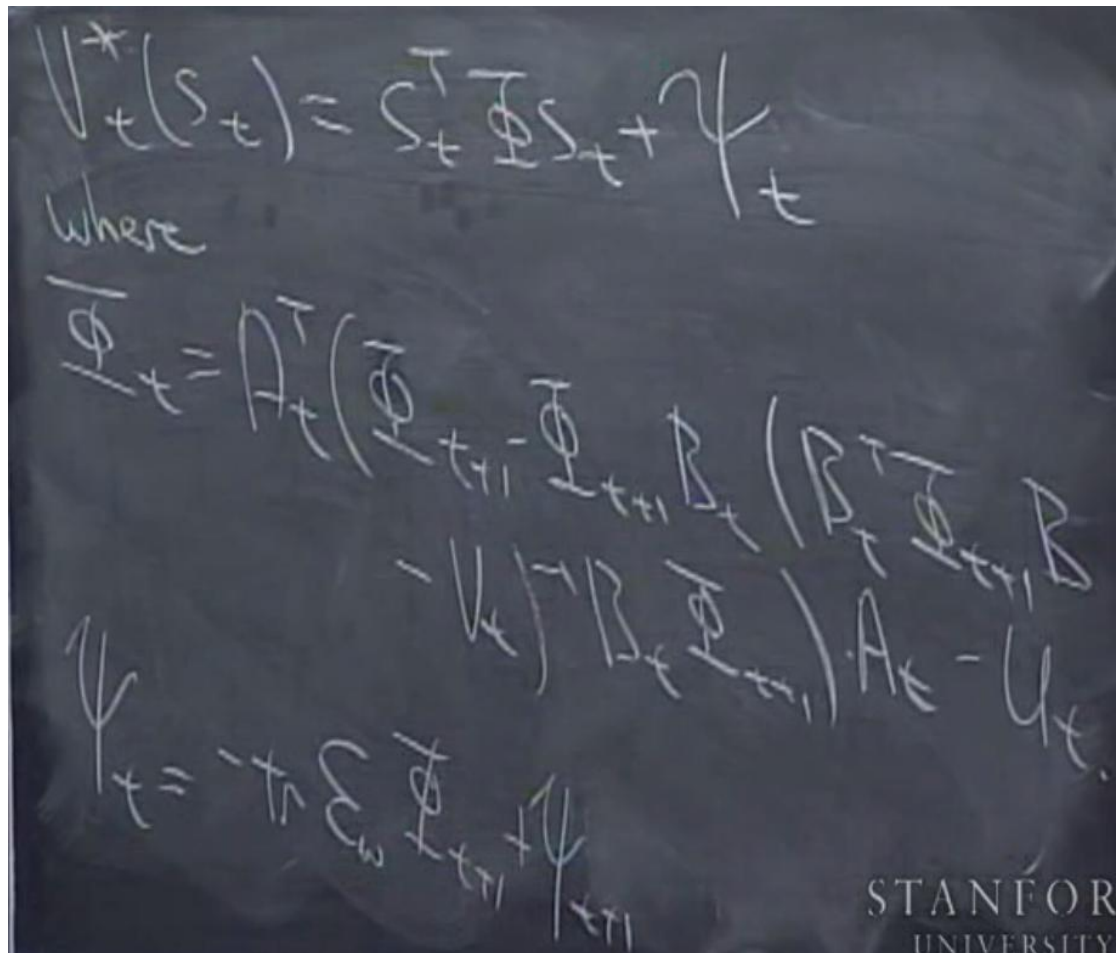
$$\pi_t^*(s_t) = \argmax_{a_t} R(s_t, a_t) + E_{s_{t+1} | N(A_t s_t + B_t a_t, \Sigma_w)} [V_{t+1}^*(s_{t+1})]$$

$$= L_t \cdot s_t$$

optimal action is linear function of current state s

optimal action is straight line

解出 a 将 a 带入上面，得到 $\Phi_t \Psi_t$ ，



Handwritten equations on a chalkboard:

$$V_t^*(s_t) = s_t^T \Phi s_t + \psi_t$$

where

$$\Phi_t = A_t^T (\Phi_{t+1} - \Phi_{t+1} B_t (B_t^T \Phi_{t+1} B_t - V_t)^{-1} B_t^T \Phi_{t+1}) A_t - Q_t$$

$$\psi_t = -\gamma \sum_{\omega} \Phi_{t+1} \psi_{t+1}$$

STANFORD UNIVERSITY

Φ_t : discrete time Riccati equation

summary:

To summarize:

Initialize $\bar{\Phi}_T = -U_T, \Psi_T = 0$

Recursive calculate $\bar{\Phi}_t, \Psi_t$ using $\bar{\Phi}_{t+1}, \Psi_{t+1}$ (for $t = T-1, T-2, \dots, 0$)

Compute L_t using $\bar{\Phi}_{t+1}, \Psi_{t+1}$

$\pi^*(s_t) = L_t s_t.$

STANFORD

19

20

