Tutorial 1

场景

我在堤坝上钓鱼。

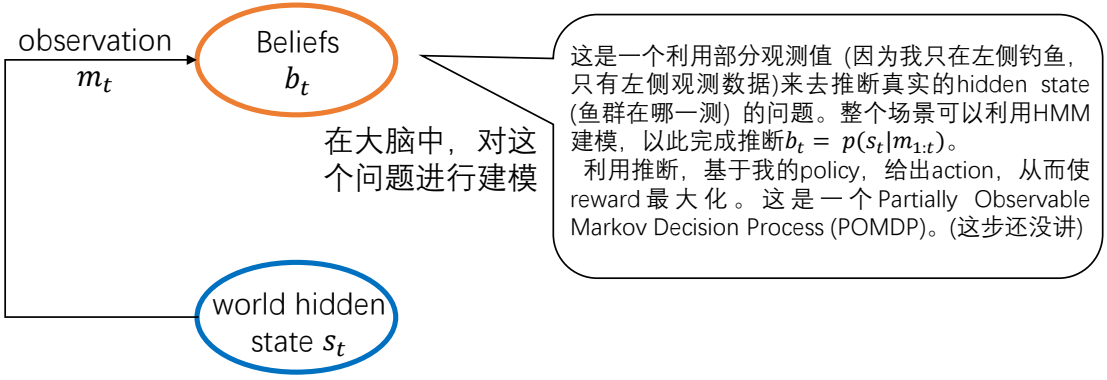
- 鱼群可以出现在堤坝的任意一侧，鱼群的换边服从telegraph process。但对于我来讲，是不知道鱼群换边的真实规律的，只是凭感觉猜测。

World hidden state s_t

- 我可以在堤坝的任意一侧钓鱼，且只选定一侧钓鱼(假设为左侧)，目前不换边。目前来讲，鱼可以在不同时刻换边，但是我不换。
  - 若我和鱼同侧，则在某一个时间点钓到鱼的概率是0.4；
  - 若我和鱼异侧，则在某一个时间点钓到鱼的概率是0.1；

Observation m_t

现在的问题是：能否利用我在左侧的观测值，即，在不同时间点上是否钓到鱼，来去推断(infer)鱼在哪一测？



场景的HMM建模

binary latent state：鱼在左侧-1、右侧+1
binary observations：抓到鱼+1、没抓到0

我在左侧(=-1)钓鱼，不换边

②状态转换矩阵A

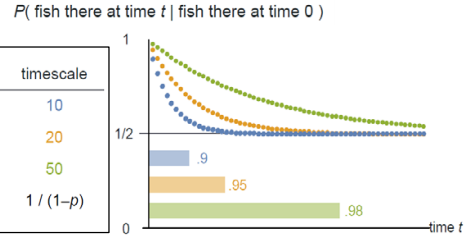
我假设鱼的dynamics服从 **Telegraph process:**

Prob(stay)=0.95

t时刻 \ t+1时刻	left	right
left	0.95	0.05
right	0.05	0.95

在0时刻，鱼在某一侧的条件下，在t时刻鱼在同一侧的概率是？可见，随着时间的增加，鱼离开该侧的几率增加，在t非常大时，鱼处于两侧的概率均等(注意，是这个条件概率)。即，t时刻在哪边不受0时刻影响。

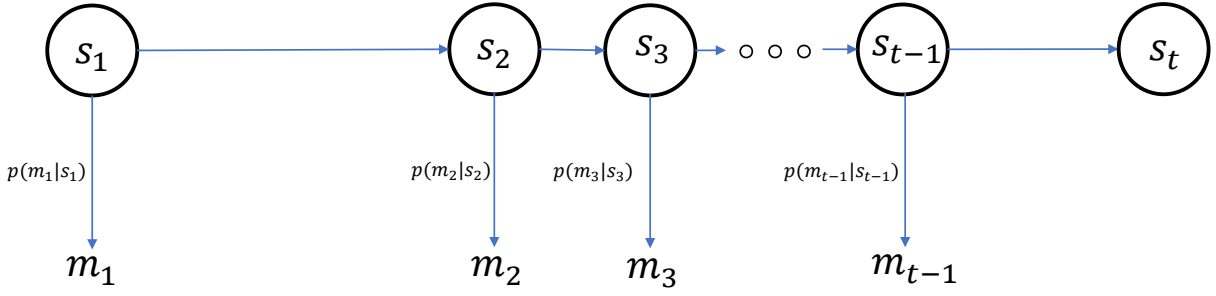
prob(stay)	timescale
0.9	10
0.95	20
0.98	50
p	$1 / (1-p)$



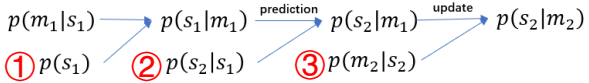
①隐变量的先验: $p(s_1)$

③Emission Matrix:

s_1 \ m_1	m_1=1	m_1=0
s_1=-1	$p(m_1\|s_1)=0.4$	0.6
s_1= 1	0.1	0.9



利用上述概率进行Inference的流程：



场景的HMM建模总结 及 inference

You only learn indirectly from what you've caught:

- This is an HMM!
  - Binary latent state
  - Binary observations (unlike DDM)

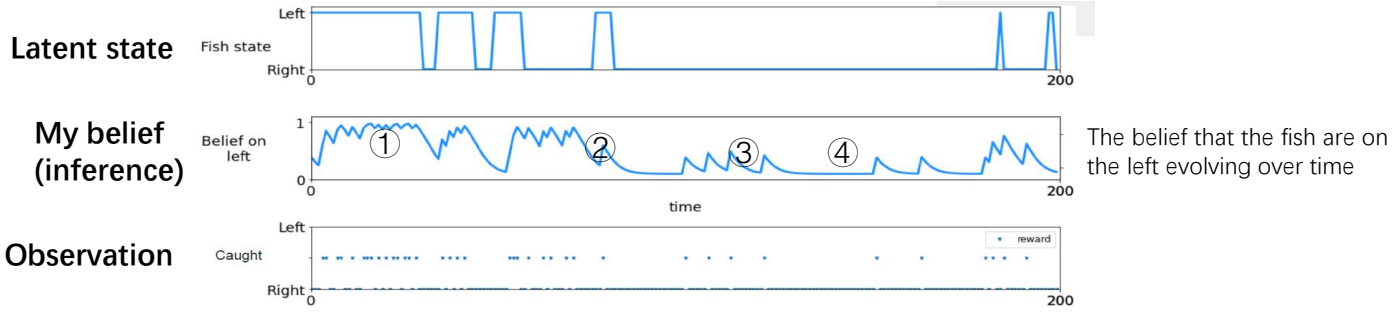
    上一节的Drift Diffusion Model (DDM)的观测值是在高斯分布中的连续采样
  - Only observe on one side

Recall HMM posterior equations:

$$p(s_t \mid m_{1:t}) = \frac{1}{Z} p(m_t \mid s_t) \sum_{s_{t-1}} p(s_t \mid s_{t-1}) p(s_{t-1} \mid m_{1:t-1})$$

This posterior quantifies our "belief" about where the fish are.

HMM inference的结果



The belief that the fish are on the left evolving over time

①when the fish are on the left side and you're catching a lot of fish, then your belief that you're in the correct place is quite high.
②But then the fish leave, you don't catch very many and your belief goes down.
③There are times later when the fish are on the other side, but still every once in a while you catch the rare fish (10% prob), and suddenly you have a glimmer of hope that maybe the fish are back on your side.
④But then no, it decays back down towards the low probability.

So now the question is: how should you act?

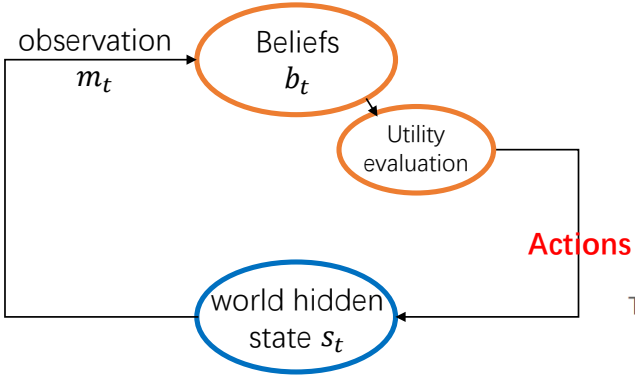
到目前为止，我们做了什么？

1. 我阐述了现象：鱼在两侧游来游去，可能具有某种dynamic；
2. 我具有观测：我钓到的鱼时多时少；
3. 提出了问题：我希望通过我的观测来infer鱼究竟在哪一边；
4. 我对鱼的dynamic进行了假设，认为是一个linear dynamic sys；
5. 基于dynamic假设、我的观测，对整个过程进行了binary HMM建模；

由此，我就对每个时刻鱼的位置进行了推断。

$$\underset{\begin{pmatrix}\text{L=0.95}\\ \text{R=0.05}\end{pmatrix}}{s_{t+1}} = \underset{\begin{array}{c}\text{left right}\\ \begin{array}{l}\text{left}\\ \text{right}\end{array}\begin{pmatrix}0.95 & 0.05\\ 0.05 & 0.95\end{pmatrix}\end{array}}{A} \underset{\begin{pmatrix}\text{L=1}\\ \text{R=0}\end{pmatrix}}{s_t}$$

利用我的推断，基于我的policy，如何采取行动以最大化我的utility (效用or获益量)?
control task: where you should be at any given time, when should you stay on one side, fish longer, and when should you switch?



Now, the only thing that we have to do is update that equation according to the fact that we are making actions that affect the world state, i.e. the state of the fish, and us, especially us. Now we have the same probabilities, but they are also conditioned on what actions we've taken. But it's still recurrence.

$$p(s_t \mid m_{1:t}, a_{1:t-1}) = \frac{1}{Z} p(m_t \mid s_t) \sum_{s_{t-1}} p(s_t \mid s_{t-1}, a_{t-1}) p(s_{t-1} \mid m_{1:t-1}, a_{1:t-2})$$

This posterior quantifies our "belief" about where the fish are.

t时刻的action基于t时刻的belief给出，来影响environment，从而影响t+1时刻的观测和belief。

Utility：在各种情况下，我的获益量

World State Utility:

抓到鱼的获益
- Utility of fish = 1

处在某个状态的期望获益
- Utility of state = Utility of fish x Probability of catching fish in state

Action Utility:

转换状态的行为带来的消耗
- Utility of staying = 0
- Utility of switching = –Travel cost

Belief State Utility (assuming accurate beliefs):

我主观认为的、处在某个状态的获益？
- Utility of fish = 1
- Utility of state = Utility of fish x Probability of catching fish in state
  x Belief that fish are on your side
  + Utility of other side x Belief that fish are on other side

And there's always the probability that you're on the other side too, which has its own utility. So you take a weighted sum of those two utilities given your probability or your belief about that.

什么意思？假设state=left，则1×0.4×我对鱼在左侧的belief 0.9，为什么还要加上the other side 1×0.1×0.1?

Policy：what to do in any situation?

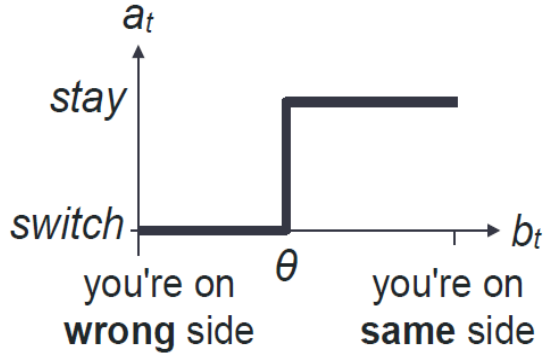
Here, the 'situation' is the current State:

- Your position s_t^{you} 在t时刻，我所处的状态
- Your belief about the fish position, $b_t^{fish} = p(s_t^{fish} = \text{Right} \mid m_{1:t}, a_{1:t-1})$
  在t时刻，我认为鱼所处的状态

There are only two actions: **stay** or **switch**.

Policy: switch when chance that you're on the same side as the fish is too low.

How low?



利用我的实际状态s_t^{you}和对鱼的状态的估计b_t^{fish}，来做出actions

Policy Evaluation

Value is the **total expected utility**

$$\text{Value} = \frac{1}{T}\sum_t U(s_t^{\text{you}}, b_t^{\text{fish}}, a_t)$$

每个时刻获得的效益utility与
- 该时刻我的状态，即，在哪边
- 我对鱼的状态的估计，即，我认为鱼在哪边
- 由此产生的行为，即，是否换边有关

例如，此时我在左侧，但是我觉得鱼已经在右侧了，因此我采取了行动，换到了右侧。
这个行为之后，我主观认为我获得的utility是多少？

	Left	Right
belief	0.2	0.8

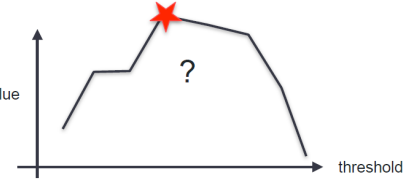
Action -> right

那么，我就认为我和鱼在同一边了，expected utility为1×0.4，但是，还不够。因为还要乘以我相信鱼在右边的程度，即1 ×0.4 ×0.8；当然，还有另一种可能，就是鱼还是在左边，但我已经换边了。这样的话， expected utility仅为1×0.1，在我看来这种情况的概率是多少？是0.2。因此，1×0.1×0.2。
综上，我基于s_t^{you}, b_t^{fish}并做出决策a_t后，我主观认为的获益量为：1 ×0.4 ×0.8 + 1×0.1×0.2。**这就是belief state utility。**

将utility和policy联系起来，我们的policy就是为了utility的最大化

What's total expected utility? There are two ways to calculate this.
- One is by using the fraction of time that you're on each side and the fraction of time you're taking actions.

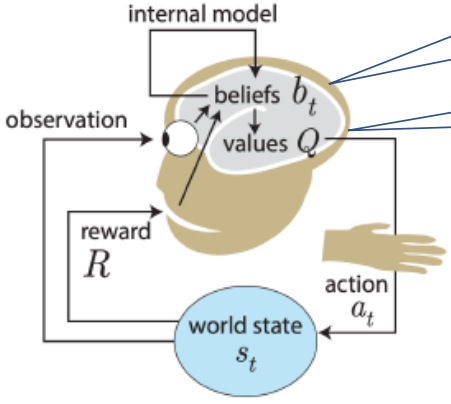
  = fraction of time on each side x fraction of time rewarded
  – switching rate x cost per switch



- But here the simpler approach is just simply to sum up the rewards that you get over time and subtract off the costs that you incur over time. So that's what we'll do here. You'll define a value function which just adds up all the fish you catch minus the costs of switching. Then you'll plot the value as a function of the threshold. you'll see some kind of curve and the optimal threshold will be the peak of that curve.

It turns out that the threshold policy that you just implemented **is in fact optimal control** for this problem if you choose the threshold well.

最后，提出 Perception-Action loop



an internal model of how the dynamics of the world work
大脑对这个动态情景的建模，就是上述的HMM过程。并由此得出了对hidden states的belief， b_t。

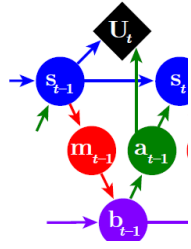
what is valuable for the agent
我的价值评价体系

take actions accordingly, to maximize my expected value

Bayesian modeling provides key tools for formalizing and understanding the perception-action loop.

Tues-Weds covered models for time series, and integrating **measurements** over time.

Thurs-Fri we're using those tools to go beyond perception and include **action**!



W2D1: 提供了大脑建模的贝叶斯基础;
W2D2~3: 基于贝叶斯，对观测变量、隐变量在大脑中进行linear dynamic的建模，并由此对隐变量进行了推断;
W2D4~5：基于推断，如何进行最优控制，最大化效益。

Tutorial 2

the optimal control of the **continuous** hidden state as opposed to the binary hidden state in tutorial 1

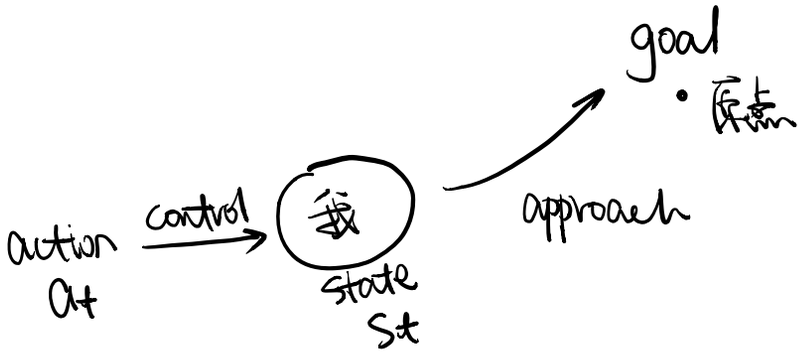
Section 0

0.1 Difference between open- and close-loop optimal control

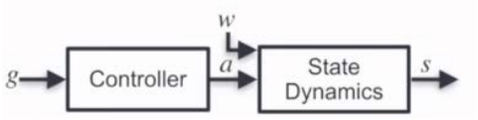
- Open loop, you solve the Partially Observable Markov Decision Process (POMDP) under the assumption that you don't ever know what the future will hold. You just have to make a guess **based on your starting point**.
- On the other hand, if you keep measuring what the world is doing as you take actions, this allows you to have closed loop control.

Section 1

You can try to get to the **goal**. The jetpack creates **an action**. So that is your **control input**. The **state** that you're trying to control is **your body position**. you want the state to be equal to the goal.



1.1 Open-loop control



g=goal; a=action; w=noise; s=state
controller 根据 goal 来生成一个 action，与噪声 w 一起作用于系统，改变系统的状态 s

State:

we'll consider firstly that **the state has some dynamics of its own**. So that means that **the state is drifting in space even without any action on it**. In general, these state dynamics can be a non-linear function. Here f is a nonlinear function of your state at the last time point and of the noise.

$$s_{t+1} = f(s_t, w_t)$$
$$s_0 = s^{init}$$

- **Hidden State 在没有 action 情况下的非线性 drifting**

Now that becomes very tricky mathematically--in this tutorial, we'll consider that your state is a linear function of the state of the last time point and an additive noise term.

linear relation　process noise

$$s_{t+1} = Ds_t + w_t$$
$$s_0 = s^{init}$$

initial condition

- **本例中假设是线性的 drifting**

  - This linear function is a good approximation for a lot of dynamics.
  - The process noise is any kind of environmental noise that may be affecting you that cannot be modeled. in this case, it could be meteorites hitting you and changing your body position in an unexpected way.
  - And we'll start off with an initial condition.

Action:

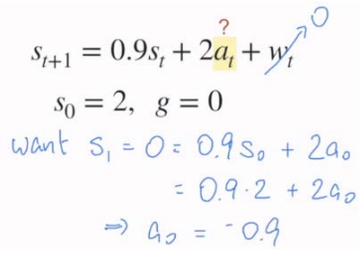
Now we apply an action in order to get to the goal. **This action also linearly adds to your state dynamics.** So, any action at time t, one unit of action at time t will create B units of change in your body position at time t plus one.

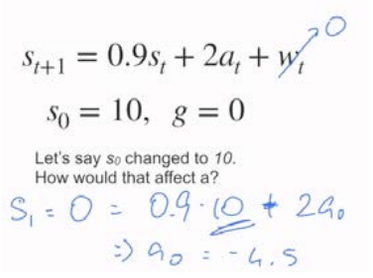
- **Action 与 state 怎么联系？这里给出了答案：两者通过线性矩阵 B 联系在一起**

$$s_{t+1} = Ds_t + Ba_t + w_t$$
$$s_0 = s^{init}$$

How do you optimally design the action a_t to reach the goal g?

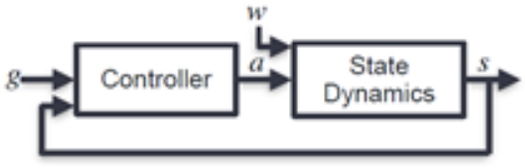
So, let's consider our goal is at the origin. let's say we would want to get to the goal in one-time step(图 1). So， a0=-0.9。What happens now if s0 actually changed to 10, so let's say you're further away from your goal, because you've been displaced further. How would that actually affect your action? you'll just need to apply a much bigger action in order to get to the goal in one-time step(图二).





Therefore, **the optimal action for one-step solution actually depends on s0, the initial condition.** But the problem is that in open loop control, in fact while we may know the goal, we don't usually know the initial state of the system s0. We may know the dynamics, but we don't know the state itself. And this creates a problem because even if you know apply a sub-optimal a_t for a couple of time steps in order to get to your goal, somehow, you are still affected perhaps by this process noise. So even if you get to your goal, you may be knocked around your goal.

1.2 Close-loop control



controller 根据 goal 及当前的状态来产生接下来的 action

- open loop control as one extreme where we may not know anything about the position
- closed loop control is the other extreme where we know the position exactly.
- Later on, we'll see a more realistic case where we have **partial observations of the state** and we still have to make an action.

Action:

our action here has access to our state at time t. In fact, we formulate it as a linear function of the state at time t, so this is the state at time t is multiplied by something called a control gain in order to get our action. So, how do we now design this control gain?

$$s_{t+1} = Ds_t + Ba_t + w_t$$
$$s_0 = s_{init}, \quad g = 0$$

Closed-Loop Control: ?
$$a_t = L_t s_t$$
Control Gain

这是本节最重要的公式：
- 首先，action 如何作用域 state？两者之间是线性关系 B；
- 在 close-loop 中，如何通过观测的 state 来生成 action？两者之间还是线性关系！由 Control Gain 连接。

综上，理解整个闭环流程是关键：
- 在没有 action 的情况下，系统自身会进行线性的、有噪声加载的 drifting

linear relation　process noise
$$s_{t+1} = Ds_t + w_t$$
$$s_0 = s^{init}$$
initial condition

这里的线性，是我们进行的假设。实际情况可能是非线性。
- 在开环情况下，加入的 action 如何与 state 相联系？两者之间线性相连。这就明确了 action 在 state 上的作用

$$s_{t+1} = Ds_t + Ba_t + w_t$$
$$s_0 = s^{init}$$
- 在闭环情况下，action 是 state 的函数，$a_t = f(s_t)$。这里假设是线性函数，由 control gain L_t决定
  因此，在闭环情况下，s_{t+1}是关于L_t的函数。
  how to design the control gain in an optimal way in order to get to our goal. our desired L0 doesn't actually depend on our initial state. This is very desirable because we don't actually have to design the controller again depending on where exactly you start from.

Section 2

if you have an initial condition that starts very far away and you still want to get to the goal in one time step, you may want to just apply **a very large action**, which would in fact be your optimal action. But it is unrealistic because machines have limitations. So, we're gonna **consider this constraint of a limited control effort**, in order to get to our goal in practice. There are two constraints:
- get to the goal,
- but use a reasonable amount of control effort.
-

2.1 Cost function

Cost function we want to minimize:
$$J(s,a) = \sum_{t=0}^{T} (s_t - g)^2 + \rho \sum_{t=0}^{T-1} a_t^2$$
- J_{state}: Get to the goal
- $J_{control}$: Limit control effort

Find a_t such that $J(s,a)$ is minimized.
$$a_t = L_t s_t$$

How does ρ dictate the solution?
- If ρ is very low or 0, the goal will be reached quickly, with little regard to the control effort.
  Quick but pricy!
- If ρ is very high, the control effort will be low at every time step, and reaching the goal will take longer.
  Cheap but slow!

Need to find optimal a_t given ρ!

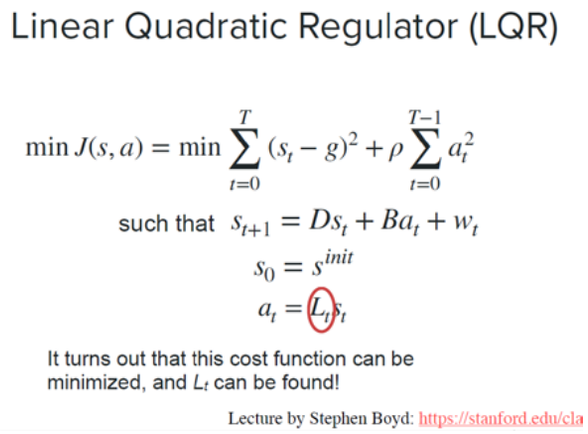
two components to the cost function
1. get to the goal and stay there
2. limiting the control effort, a_t.

in general, we want to minimize this cost function. variable ρ dictates how much we care about limiting the control effort as opposed to not being at the goal.

理解 cost function：
- 首先，这是一个累积 T 个时刻的 cost。
- 第一项：在从 0 到 T 时刻的过程中，每个时刻系统的状态s_t和 fixed goal 之间的差的平方，记为该时刻的 cost。- 第二项：我们要限制每个时刻 action 不能过大。
- 从总体来讲，action 过大，第一项随着 T 的增大迅速减小，这是我们想要的；但第二项会迅速增大，这是我们不想要的。
- 因此，基于此 cost function 寻找最优的 action，or 最优的 control gain L_t的过程中，要权衡两项，才能得出最优解。

our task: given a ρ, find an optimal a_t or optimal control gain actually. 不同的ρ，会有不同的 optimal control

2.2 Optimization

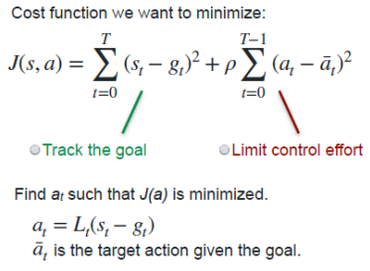


在 1) latent state 服从我们设定的 dynamics; 2) action 是关于 state 的线性函数 的情况下，来对 cost function 求极小值。this problem in general is called the linear quadratic regulator LQR. **we want to find Lt, the control gain such that this cost function is minimized.** In fact, it has a straightforward solution as this cost function can be minimized, Lt can be found using principles of dynamic programming.

Section 3

3.1 The goal is moving

how would you design the action to track a moving goal g(t)?



当 goal 实际在移动的情况下，上述 cost function 要做出什么改变？
1) Goal 是时间 t 的函数，所以在求每一个时间点的 cost 时，要考虑时变的 goal；
2) 在之前的假设中，goal 在远点，所以a_t仅仅与s_t呈线性关系。但现在，goal 是变化的，我们要根据当前自身位置与 goal 的位置的差来确定 action；
3) $\bar{a}_t$ ？

Section 4

Closed Loop Control with Partial Observations

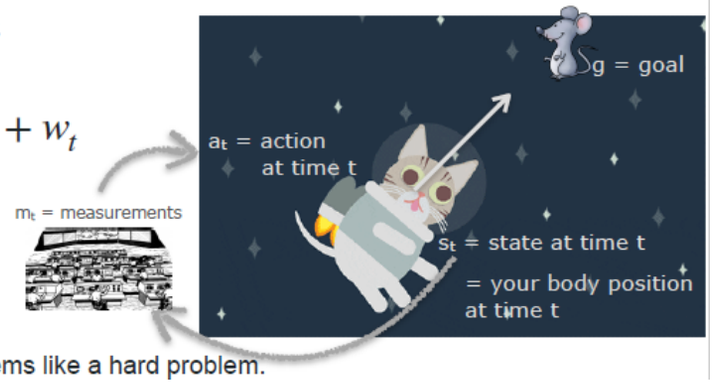
$$\min J(a) = \min \sum_{t=0}^{T} (s_t - g)^2 + \rho \sum_{t=0}^{T-1} a_t^2$$

such that $s_{t+1} = Ds_t + Ba_t + w_t$

$$m_t = Cs_t + v_t$$

$$s_0 = s^{init}$$

$$a_t = f(\mathbf{m})$$



This seems like a hard problem.

- **partial measurements:** the dynamics now include this m_t, which is a linear function of our state s_t, but C basically signifies a partial observation of that state. we will not be able to see the state entirely.
- now a_t **is a function not of** s_t **but of** m_t. So, a depends only on these measurements. How would we design a_t?

in general, this can be a hard problem 1) if this is not a quadratic function or 2) if these dynamics are not linear. But in fact, in this case, we can make use of something called **the separation principle**, which basically states that we can decompose our problem into two separate problems.

Separation Principle

In fact, the optimal policy consists of:

Linear Quadratic Gaussian (LQG)!

1. Estimating the state from noisy measurements

Kalman Filter!

2. Using the state estimate for designing the control

LQR!

hidden state

Kalman Filter Part:

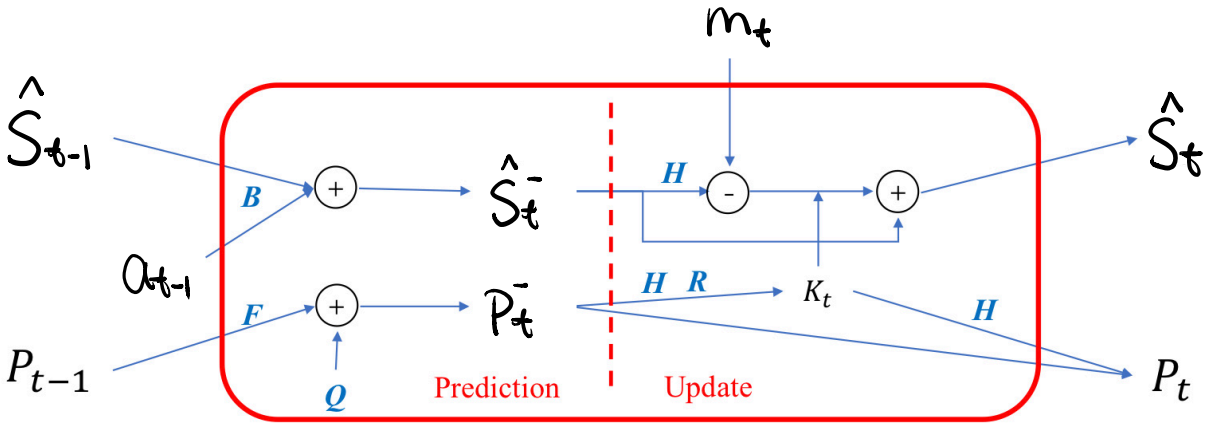
Hidden state: $S_{t+1} = DS_t + BA_t + w_t$

$A_t = L_t \cdot S_t$

$\Rightarrow S_{t+1} = DS_t + L_t \cdot S_t + w_t = (D + L_t) \cdot S_t + w_t$

答：kalman filter中，对变量连续、线性变化的假设.

利用 measurement m_t，来对 $\hat{S}_t$ 进行 inference.



可见，这五个矩阵的设置至关重要

***F*, *B*：** 由系统相关的理论、场景决定

***Q*：** 引入系统预测的不确定性

***H*：** 观测值与状态的固有关联

***R*：** 观测值的噪声衡量

trained 卡尔曼(线性)滤波器

LQG Part：

$$\min J(a) = \min\left(\sum_{t=0}^{T} (\hat{s}_t - g)^2 + \rho \sum_{t=0}^{T-1} a_t^2\right)$$

$\Rightarrow$ optimal a_t，然后重复上述过程.