LECTURE OUTLINE

[Music] okay is a new room and a new direction somewhat different direction in our class

新的教室和新的方向，与我们之前的课程不同的方向

in the first week the first three lectures were primarily directed towards background and overview

第一周的前三节课我们讲了背景知识和综述

background on exact dynamic programming with an emphasis on discounted infinite horizon problems

精确动态规划，重点讲了无限期折扣问题

and in the third lecture an overview of approximate dynamic programming which is our subject

第三次课对我们的课程涉及的近似动态规划做了一个综述

in this week these three lectures today Wednesday and Friday I'm going to focus selectively on various topics of approximate dynamic programming

这周的三天的课程我会有选择性地讲一下近似动态规划的这么多话题

we are going to go into a little bit more detail but we will by no means exhaust

我们会讲更细节的东西但是没法讲太多

the subject we will be very selective we'll pick some of the important topics and go a little bit in depth into them to give you an idea of how the field looks like it

我们会有选择性地选择重要的话题并且深入讲解，让你知道这个领域是什么样子的

will also give you a start if you are interested to go more deeply you can get started and expand your understanding and background on the subject

我会带你开始进行研究，如果你想深入地研究，就可以扩展你对你喜欢的主题的理解程度和背景知识

so what we'll spend some time and review a few slides

我会用几页带你回顾一下之前讲过的内容

and then the focus of today's lectures are approximate value iteration and policy direction for discounted problems

然后会专注于今天的课程，折扣问题的近似值迭代和近似策略迭代

we mentioned earlier that exact value duration and the exact policy duration are the two main algorithms for infinite variety of discount dynamic programming

我之前提到的精确值迭代和精确策略迭代是无限期折扣动态规划诸多算法中主要的两个

now we are going to introduce approximations into value and policy duration using the idea of approximation in value space

现在我要介绍用值空间的近似思路对值和策略的近似

the approximation value space means that I have a function and I approximate this over a lower dimensional space we talked about how you can do this by simulation that's the idea we're going to be using here for different functions like the cost function or cost function of the problem or the cost function associated with a policy and so on

值空间近似的意思是有一个函数，我想要在更低维度的空间内近似它，我们之前讨论过如何用仿真来近似，同样会用到这里，我们还会用仿真对不同的函数，比如成本函数或者问题的最优函数或者某个策略的成本函数之类的

so I think I mentioned the projected bellman equation we're going to focus into that and see how we can solve it approximately by using a matrix form of the equation

我之前提到过投影bellman方程，我要深入讨论该如何使用矩阵形式的方程近似求解

how do you deal with the large inner products in matrix vector products that are involved in this projected equation by using simulation

这些投影方程中的矩阵和向量计算时会产生大量的内积运算，我们要讨论如何使用仿真来处理这些计算需求

Vietnam where you talk about a couple of specific methods that are central methods in the field the LSTD and the LSPE method these are simulation based methods for solving the projected equation

我们之前讨论过几个很重要的方法，LSTD和LSPE，这些都是基于仿真求解投影方程的方法

optimistic version of policy duration

策略迭代的最优化方法

then some more advanced topics on multi-step projected equations and things related to that

一些关于多阶段投影方程很有用的主题和与他相关的内容

so a few slides we'll review and you may interrupt me if there's some question that's burning question but that you'd like to bring up

下面是我们的回顾，如果你有什么问题没明白可以打断我

DISCOUNTED MDP

it's a smaller group now so we can do that

听课的人比较少，所以可以这样做(打断我)

we have a finite state system a Markov chain with states 1 up to n and the finite control set u of I at state I

有一个有限状态系统的马尔科夫链，状态从1到n，有限控制集合U(i)

the chain is described by transition probabilities whereby if you are at state I in you apply control u you go to state J with probability pIJ of U

马尔科夫链被状态转移概率描述，如果你现在在状态i，执行了控制u，你会以概率P\_{ij}(u)转移到状态j

so it's a Markov chain if you fix the policy if you fix these controls it's a Markov chain but it's a control Markov chain in the sense that we want to find the good values of U and the policy that produce good effects in terms of a cost function

如果你调整策略和这些控制，这就是一个马尔科夫链，但是控制一个马尔科夫链是想要找到一个好的控制价值和相应的策略带来比较好的成本函数

so give a policy a sequence of functions at every time step PI we can plug it into the cost function which involves a cost at every stage K discounted by a factor alpha to the K with alpha between 0 & 1 take the expected value of that the limit as n goes to infinity an infinite number of stages and that gives you a number which is the cost of policy pi starting at state I

所以给定了每一个时间阶段的策略序列函数pi，我们可以把他放到成本函数中，成本函数包括每一个阶段k的折扣成本，就是成本被0到1之间的折扣因子alpha^k影响，计算当n趋于无穷时候的期望，会得到这个策略从状态i开始的成本

so for every I for every state there is a cost and is a vector of cost and dimensional vector of cost as associate with a policy

所以对于每一个状态i，都有一个相应的策略成本向量

and we recall that we used a lot the shorthand notation which is TJ for a given vector J TJ is another vector and dimensional vector with components from 1 to N given by this expression this part is the expected cost of the first stage this part is the expected cost of what you will get at the end of the first stage measured by J ok

我们回忆一下如何用速记符号来表示，给定向量J，TJ也是一个向量，维度是n，左边这项表示当前阶段的成本期望，右边这项表示从下一阶段到终止阶段的期望成本，这个后续期望成本用J表示

so you can view this cost of the first stage cost at the end of one stage ok and the minute this is TJ and T nu J and we have seen that is T mu J and TJ are central in expressing the theoretical results

所以你可以看到第一个阶段的成本和后续成本，这时候有两个概念，TJ和T\_muJ，这是速记理论中的核心概念

“SHORTHAND” THEORY – A SUMMARY

in particular the optimal cost vector is the unique solution of this bellman equation or equivalently the functional equation is nonlinear n equations with n unknowns

特别是最优成本向量，是bellman方程的唯一解，而且这个bellman方程是非线性方程，有n个方程和n个未知数

and for a fixed policy is a corresponding bellman equation again n equations with n unknowns but it's linear okay it is linear

一个n个方程和n个未知数的bellman方程组，有一个不动策略(参考fix point翻译的)，这个方程组是线性方程组

and therefore it can be solved by by standard methods

因此可以使用标准方法求解

the optimality condition is but if I can find J star then by minimizing in this equation by minimizing in the right hand side the that equation I can get an optimal policy in reverse every optimal policy has to satisfy has to minimize in the right-hand side

这是最优条件，我可以通过最小化这个方程来求J\*，最小化右边这个表达式计算最优策略，反过来，最优策略一定满足这个最小化条件

THE TWO MAIN ALGORITHMS: VI AND PI

and the two basic Alcazar value duration whereby I start from some J vector arbitrary J vector and iterate with it with T do a dynamic programming step on it then another one and another one sequentially and in the limit I'm guaranteed to get J star the optimal cost vector that's value direction

这两种基本算法，从一个任意的向量J开始，每次迭代都使用动态规划算子T，在迭代次数趋于无穷的时候，可以保证得到最优成本函数J\*，这就是值迭代

in policy Direction iterates with policies at the typical step the case step I have a policy which is a control to apply at every step I evaluate the policy by solving the corresponding bellman equation were in shorthand like this after I evaluate it I plug it in here and I get a new policy by minimization and in shortcut is this

策略迭代中从一个策略开始，通过求解这个bellman方程评估这个策略(policy evaluation那个表达式)，这个是速记符表示的表达式，然后把他放到这个表达式中(policy improvement)求解获得新的策略，下面是速记表达

now the policy evaluation is equivalent to solving a linear system of equations and by n equations

新策略的评估等价于求解一个n个方程的线性方程组

but we are interested in huge dimensions and extremely large so you would never think of solving this equation by Gaussian elimination or giving into MATLAB or whatever you have to use a different kind of approach

但是我们一般会求解的是高维，非常大规模的方程组，所以我们从来不考虑通过高斯消元或者matlab来求解，必须要用其他方法来解决这个问题

and one possibility is to try a finite number of of well one possibility is to use approximations and we will see how approximations come into the picture for evaluating for solving approximately this equation okay

一种可能是尝试使用有限次，哦，一种可能是使用近似方法来求解，一会我们就会看到如何使用近似方法求解这个评价方程

APPROXIMATION IN VALUE SPACE

so now I may approach these optimist approximation in value specs

这个方法是在值空间中找一个最好的近似方法

there are these objects of interest J star in J\_mu

这是我们求解的目标J\*和J\_mu

and we want to find something that's lower dimensional is described by fewer numbers

现在我们想要的是更低维度的方法

and in particular were interested in a function J tilde that's a function of state but also depends on some parameter vector R

这是一个状态的函数J tilde，这个函数的值还依赖于参数向量r

and R is a vector with M components actually s components should be yes

向量r有m个元素，说他有s个元素也对

okay a type but a small number of components which you can tune you can adjust to achieve a better approximation

这个向量元素数量很少，你可以调整他的值得到一个更好的近似函数

so think of M be huge but s the number of parameters being small and there are many types of different approximation architectures that I can use polynomial approximations interpolations domain-specific like the tetris problem i discussed last time

一般认为m很大s很小，有很多种近似结构可以用，比如多项式近似，插值近似，特定域近似，比如我最后一次讲的俄罗斯方块

depending on the problem you would choose a parametric for a general functional form of dependence of J tilde on iron r

依赖于问题你可以选择一个通用形式的函数J tilde

and the objective is to get a good R okay in what sense

某种程度上，这时候的目标是找到一个比较好的r

well any r defines some function here and if this is a good approximation to j\* then I hope that my minimizing in this approximate form of the right hand side of bellman equation I will get a good good suboptimal policy new tilde

r被定义成这样，如果这是一个比较好的对J\*的近似，我希望最小化这个表达式(中间那个很长的表达式)获得一个比较好的次优策略mu tilde

so the whole game is to find a good r good in the sense that it gives me a good muted

所以整个任务就是找到一个比较好的r，找到之后就可以获得比较好的近似策略mu tilde

now we are going to focus in this in this lecture exclusively on linear architectures

在这个课程中我们会一直使用线性结构

where the of J tilde on r is linear okay so this is a linear for a given J this is a linear function

也就是基于r的近似函数J tilde是一个线性函数，所以对于一个给定的r，J是一个线性函数

and more generally the vector of these is a matrix a gigantic row matrix or a dimensional matrix and small column dimension multiplying R which is a small dimensional vector

更一般地，这是一个行多列少的矩阵，这个矩阵乘以r可以得到一个低维度的向量

so this is a huge vector but R is a small vector and phi is a matrix of basis functions

所以这是一个很大的向量(指J tilde)，r是一个低维向量，phi是一个基函数输出的矩阵

LINEAR APPROXIMATION ARCHITECTURES

in particular let's recall what a linear approximation architecture is

我们来回顾一下线性近似结构是什么样的

for every state I there is a vector Phi sub I ok small number of terms which we view as the features of the state I

对于每一个状态i，都有一个项非常少的向量phi(i)，可以叫他状态i的特征

and this is these features are weighted with a vector R and this gives you the approximation form

这些特征被向量r加权，累加之后就是近似的形式了

Phi I prime is the I-th row of fee so fee has Phi 1 prime Phi 2 prime Phi 3 and so on and n of those

Phi(i) prime 是矩阵phi的第i行(i可以理解成状态i的索引)，所以有phi 1 prime Phi 1 prime Phi 2 prime Phi 3什么的，一共有n行(n是状态的总数量)

and alternatively you can write this entire vector as a linear combination of the columns of fee okay weighted by the corresponding coefficients of R

作为替代，你可以把近似函数写成矩阵phi的列的线性组合的形式，向量r对矩阵phi的各列进行加权

so if I give you R how do you get an approximate evaluation of whatever you are approximating this J tilde

所以我给你r以后，你怎么能获得一个近似函数J tilde的值呢

for a given state whenever a state comes to you you extract its features numbers that describe in some broth where a rough but effective way the state so you get this vector Phi sub I this is a small vector okay and you weigh this with the small vector R and you get a prop cost approximator

给定一个状态后，你从状态中提取特征，然后计算向量phi(i)，这是一个低维的向量，然后用向量r对他们进行加权求和就得到了近似成本

so if you have this linear map this r then and you have congest good features you're in business

如果你有一个线性映射r还有一个比较好的特征提取方式，那你就成功了

okay now another view of approximation is based on this equation

另一种近似是基于这个方程的(J tilde那个近似形式)

what this says is that we are looking for approximations of whatever in the subspace Phi R where R lives in RS is a vector in R s and this is the the subspace spanned by the columns of field

我们在做近似的时候，使用的是phi r的子空间，r在R^s空间中，也就是说，这个子空间是phi的列的线性空间

so J whatever you are approximating lives in a gigantic space okay but there's a smaller space defined by these features and what we want to do is approximate it down into the smaller space

所以你想要近似的J在一个非常大的空间内，但是我们用一个被特征矩阵定义的更小的空间来进行近似，也就是我们想要用更小的空间来近似完整的空间

so that's the idea instead of computing j mu or j star which is huge dimensional we compute the lower dimensional r vector by using lower dimensional calculations okay

这种近似J\_mu或者J\*的想法就是通过在低维度空间内计算低维度向量r来代替高纬度空间内的计算

low dimensional calculations is not so easy to do okay you're going to collect information about huge dimensional stuff in low dimensional calculations that's where simulation becomes important that's what simulation will do for us will allow us to obtain low dimensional approximation with low dimensional calculations

低维度空间的计算也不容易，我们收集高纬度空间的信息映射到低纬度空间内，这时候仿真就变得很重要，因为仿真让我们可以进行低纬度空间的运算

APPROXIMATE VALUE ITERATION

okay so now we will consider the two basic methods value direction and policy direction and make a first attempt at introducing approximations into them

现在我们要讲两个基础的方法，值迭代和策略迭代，我要进行第一个尝试，介绍一个他们的近似方法

APPROXIMATE (FITTED) VI

now this okay this approximate value direction sometimes is called also fitted value direction you'll find this term in the literature and what it does is very simple

近似值迭代有时候也被叫做拟合值迭代，你可以在文献中看到这种说法，实际上很简单

value iteration calculates J(0) TJ0 T squared of J 0 and so on

值迭代计算J0，TJ0，T^2J0之类的(应该是想说T^kJ0)

so at every step of this algorithm we introduce an approximation which involves some kind of a projection to a lower dimensional space and keep going

这个算法的每一个阶段我们都会介绍一种从高维空间投影到低维空间的近似方法

so the initial function J 0 is given may be some value R 0 and here's the algorithm we started J 0 I have the vector T sub J 0 and I find some kind of approximation onto the lower dimensional subspace of that so J tilde sub 1 is an approximation of TJ 0 now I apply T to it again then I approximate it down the lower dimensional space again and so on can i generate a sequence of lower dimensional vectors R 0 here R 1 R 2 and so on

初始化J0和r0，这个算法是这样的，从J0开始，我使用算子T获得TJ0，然后在低维空间使用J tilde 1近似TJ0，然后使用T获得TJ tilde 1，再在低维空间使用J tilde 2近似TJ tilde 1，就这么一直进行下去，结束的时候我就可以获得一个序列的低维向量r0，r1，r2…

okay if I were to do value iteration I've start at this J 0 I would go to TJ 0 T squared J 0 T cube J 0 and so on

如果我用值迭代的话，我从J0开始，进行T^2J0，T^3J0…的操作

here i generate T sub J 0 approximate that then another T approximate that another T approximate that okay

现在我做的是生成TJ0，近似他，然后再使用T，再近似他，继续使用T，继续近似他

that's called fitted value directions are very very natural algorithm I have an iterative algorithm i approximate every step of the algorithm in a lower space

这就被叫做拟合值迭代，非常自然的算法，迭代算法在每一次迭代中都在低维空间进行近似

then after I do a large enough number of iterations let's say capital n the final J tilde is going to be an approximation to whatever approximating J star here in particular

如果迭代足够多的次数，比如N次，最后得到的J tilde就会很接近你想要近似的值，比如J\*

this can be used with both T equal to the minimization mapping or for a specific policy can be used

这个算法最小化映射T和特殊策略都可以使用

and how do I find this J tilde approximation from this well one possibility is to use some kind of approximate projection done with respect to some projection norm typically our Euclidean norm although in principle doesn't have to so approximately what I'm doing is from J tilde K I apply T and then I project it down and that gives me at least approximately J tilde K plus 1

我该怎么找到近似的J tilde呢，一种方法是用某种形式的近似投影，通常用欧几里得范数，虽然原则上不是必须这么近似，我正在做的就是从J tilde k使用算子T然后把结果投影到低维空间然后得到近似的J tilde k+1

it's an old algorithm very old idea perhaps it goes back to the to the s or even s people have thought of doing this it's very natural however you'll see that it's not as simple as it appears

这是一个非常古老的算法，可以追溯到五六十年代，人们都认为这种算法很自然但是你可以看到，它并没有看起来这么简单

okay so since we are into the subject of projections let's look let me remind you what we mean by projection

我们要讲投影的细节，我要告诉你我说的投影是指什么

WEIGHTED EUCLIDEAN PROJECTIONS

in the particular the projections for today are not going to be with respect to a Euclidian norm however with some weights sigh okay

今天讲的投影，都是带有权重xi的欧几里得范数

so if xi is equal to 1 this is the standard norm square root of the sum of the squares

如果xi等于1，这就是一个标准范数的平方和的平方根

however we introduce xi vector here with positive components in fact there is no difference there's not without loss of generality I assume that's a probability distribution the xi vector is a probability distribution with positive components

这里的向量xi是一个正向量，事实上，不失一般性，可以假设它是一个概率分布

and I'm going to consider our weighted norm and I can tell you ahead of time that's important to consider why it's here ok

我要把它当成一个加权范数，一会我会告诉你为什么一定要把它当成一个加权范数

now let's consider projection onto this subspace with respect to this weighted norm

我们来看一下这个带有加权范数子空间的投影

in other words we want to find and r vector of parameters such that Phi sub R star is equal to the projection

换句话，我想要找到一个参数向量r，让phi r\*等于这个投影

how do we do that well I want to find within this subspace an approximation in this sense of the weighted norm of J so given J r star is solved by solving this what this this quadratic problem shall be squared problem

那么我们如何找到这个r呢，给定J的时候求解这个二次问题(红字上面的那个表达式)就可以了

however the problem the difficult is the following

这个问题的难点是这样的

J in our context it's a huge dimensional vector so I can't possibly compute it and then try to approximate it ok

在我们的问题中J是一个非常高维的向量，所以我们不可能执行与他相关的计算，只能尝试近似这个结果

so instead I mentioned in the previous lecture that we may use simulation how the simulation go

所以我之前讲过这个内容，可以使用仿真来计算，那么如何使用仿真呢

I compute some values for some states of this

我计算J中一部分状态对应的值

may be my states are let's say 1 million and I compute J sub I for 10000 states ok which I generate somehow by sampling at sample the state space I calculate this and then I can make a least squares fit that is not as computationally intensive ok so that's how simulation goes

状态空间可能有一百万个状态，但是我只通过采样获取一万个状态，只用这一万个状态计算这个问题，就可以避免大量的计算了，仿真就是这样工作的

and in particular I can there's a formula that I gave last time

这是我最后一次写这个表达式(本页最后一个表达式)

that the implementation by simulation of this projection with respect to this norm is done as follows

执行仿真进行投影范数的计算是这样做的

we sample J sub I for a set of states I 1 I 2 and so on and then i saw i restricted least squares problem

对状态进行采样得到i1，i2…然后得到J(i)，然后用这些数据限制这个最小二乘问题

ok this is the same as ok you can do it in two ways approximate an expected value or solve a corresponding problem

你可以使用两种方法来做这件事，对这个值进行近似，或者解这个最小二乘问题

however notice the following

注意我下面要讲的内容

that the samples have to be collected according to the distribution xi okay

采样必须在分布xi下进行

because if you sample according to a certain distribution the corresponding terms involving I will appear with different frequencies the frequencies corresponding to sigh that's why the solution to this approximates the least square solution with respect to the sign norm ok

如果你用一个确定的分布采样，相关的i会出现的频率不一样，这个频率是与分布xi相关的，这就是为什么近似最小二乘问题的范数与xi有关

so what do I gain here

那么这里我能获得什么呢

first of all no gigantic calculations I just need to be able to generate samples perhaps approximate ok samples of some point

首先，我可以不用做大量的计算了，只需要生成样本就可以进行近似了

and second the other thing that I gain is that is that I am able to control the norm by sampling according to the corresponding distribution

第二个是我可以通过在某分布下采样控制范数的值

that's an important idea for our context low dimensional calculations can control the projection approximation through sampling

这是一个非常重要的想法，在低维空间计算可以通过采样控制投影近似的结果

(Question time

试错，通过置信区间调整采样策略(prof不认为这玩意好用)

选择分布xi，今天会讲

any questions yes how can we guarantee the accuracy well in practice it's more of a trial and error process you can get of course some kind of confidence intervals here but they're not very useful in practice I don't think okay perhaps they are useful but I don't think that's the main issue how do you get the accuracy you just try to get as many samples as you can and then your accuracy improves as much as possible okay as many samples as you can give them the computing time that we can afford you mean the the the Phi matrix okay this is very much problem dependent and also it is a separate decision okay you there are two issues first select the approximation subspace then you then you try to solve try to find an approximation with the subspace the two are different of course if you are not satisfied with the results that you have you may go back in may add some more features secret kind of features there may be a process of iteration between approximation and RT and in modifying the approximation architecture - we fix the distribution to decide okay you okay so if you know the distribution you can incorporate it into the sampling now how you choose the sky is a major topic for today's lecture okay so I'm going to get to that okay question finished)

FITTED VI - NAIVE IMPLEMENTATION

so remember we are talking about this fitted value direction successive approximation and now let's try to use projection

还记得我之前讲的拟合值迭代么，它能很成功地进行近似，现在我们尝试一下投影

here here's a naive implementation of fitted value direction

这是一种朴素地拟合值迭代算法

I have a large number of state a large state space let's select a small subset of representative states okay I\_k I call this subset

有一个很大的状态空间，我要选择一个有代表性的子空间I\_k

and for every state in this subset calculate this expression here

用这个集合中的所有状态计算这个表达式的值

okay so instead of calculating for all states you calculate only for the small subset

相比于对整个状态空间的所有状态，你只使用了状态空间的集合来进行计算

now how do you calculate this okay

你要怎么进行计算呢

so this is an expected value here

等号右边的表达式是一个期望值

this is given okay this is given

J\_k tilde已经给定了

this is an expected value

整个表达式是计算期望值

so likely you'll do this by simulation

所以你可以使用仿真来求解

okay and you'll do it for every u and then you have to minimize over on u

在计算的时候是对于所有的u来说的，所以你需要在所有u里最小化

so we're talking about a substantial calculation here for every I but at least you don't do it for all the states are you doing only for a subset

所以我们在谈论包括每一个状态的大量计算时，我可以只对一部分状态进行计算

so these are the samples that we compute

这就是我计算要用到的样本(I\_k)

and now we fit the samples to the new iterate

所以我在每次迭代时拟合这些样本

okay so I have a small set of values here

所以现在我有一个比较小的集合((T tilde J\_k)(i))

and I use a form of approximate projection to fit a new function J tilde k plus 1

然后我用投影近似去获得一个新的函数J\_{k+1}

and if you use simulations here then you don't need a model by the way okay you don't need these transition probabilities if you can simulate them by some the computer

如果你用仿真计算的话就不需要知道模型了，如果用仿真来求解的话就不需要知道状态转移概率

and there is some error bound that theoretical error bound and that you can compute for this process

这个计算过程是可以算出误差上界的

and this error bound says that if okay let me go back to this figure here

我们先回到上一页

if this difference is within Delta okay if these differences are within Delta so presumably you have chosen intelligently enough the subspace so that the iterates do not differ are not too far away from this subspace

这个差值与delta有关所以你可以足够智能地选择一个与子空间不是那么远的解

if this differences are within Delta then in the limit of this process of successive fitting you get a function which is within this quantity of the optimum for all okay for all I so this looks somewhat nice okay

这个误差与delta有关，连续拟合过程迭代的极限可以得到一个每一个状态距离最优解最多差这么多的函数，这个结果看起来还是不错的

that is some guarantee even though this this alpha here which appears squared 1 minus off in the denominator which is a little discouraging nonetheless it's not as if okay some result

分母上的(1-alpha)^2能够保证误差不会太大，但是这并不是一个非常好的结果

however there is a potential problem with this and I'll show you what the problem is in the next slide

这是一个潜在的问题，我会在下一页给你展示这个问题

because it may be that you cannot guarantee that this holds uniformly there's a delta that holds uniformly for all K

这个问题就是你无法保证对于所有的k，delta都能保证误差均匀

AN EXAMPLE OF FAILURE

okay here's an example a very simple example what is this algorithm fails completely

这是一个非常简单的让这个算法完全失效的例子

suppose that you have just two states okay Stage one and two okay two states and there's only one policy so let's forget about minimization there's only one policy we just try to approximate the cost function of that policy

假设这个问题只有两个状态1和2，只有一个策略，所以不需要做最小化了，只有一个策略

我们只需要对这个策略的成本函数进行近似

now let's assume that this Markov chain from State one goes directly to state two that and if it is that state two just state that state two

假设这个马尔科夫链从状态1直接跳转到状态2，或者状态2跳转到状态2

you can't get any simpler than this okay two states deterministically from wanted to or stay within two and with no cost okay

这是一个没法更简单的系统了，从状态1确定性地跳转到状态2，或者从呆在状态2不变并且不产生成本

so trivially the cost function as the cost vector associated with its Markov chain is zero and zero there's no cost okay your humilate zero at every step no matter what you do well there's only one thing to do but your accumulator

马尔科夫链的成本函数是零向量，无论你在每个阶段做什么，成本都是零，这个系统中你只能做一件事请

so my desire object is the zero vector and i want to approximate it within some subspace and i will choose the sub space that is spanned by this vector here

现在解是一个零向量，我想要用子空间S和这个向量xi近似他

so let's consider the value iteration scheme that approximates that approximates the tracks approximation start with in this subspace

所以考虑使用值迭代理论在子空间中进行近似

so this subspace is described by a single parameter R

所以这个子空间被一个向量r表示

and the subspace are all the vectors that are multiples of one and two okay

子空间就被这个向量(1,2)表示

so J star is 0 it is a subspace here spanned by a single line

所以J\*是0，子空间被一条直线穿过

now 0 belongs to the subspace so you have here even the luxury that J star the object that you're looking for belongs can be approximated exactly okay with zero error

so it's a favorable case

这是一个很有利的情况

now let's see how this fitted value duration scheme is going to work

现在我们要看看拟合值迭代是怎么工作的

what do we do well we have a vector in the approximation subspace a vector of this line defined by RK

我们有一个近似子空间的向量，也就是这条被r\_k定义的直线

and we find a new vector on this line according to this projection

我们可以通过对TJ\_k进行投影得到直线上的新向量

which I be the weight of the projection though the norm

通过求范数与加权获得新向量

so that's that's that's the fitted value direction

这就是拟合值迭代

I have a line I have some point in the line and I go to another point in the line another point in the line and ideally you'd like to go to zero

有一条支线和这条直线上的某些点，然后迭代到支线上的另外一些点，再迭代到另外一些点，理想化的结果是你能够迭代到0

so how do you do that well you approximate this vector R two R you try to find a least squares fit to these values here now this values are RK + 2 RK so it's a very simple calculation here it's a linear least squares problem with a single parameter

你近似这个向量，r和2r，想要找到一个最小二乘问题的解，现在TJ tilde(1)的值是r\_k，TJ tilde(2)的值是r\_{k+1}，这个计算非常简单，是一个只有一个参数的线性最小二乘问题

I'm not going to go to the mechanics of solving this problem I'll just give you the solution

我不打算给你看求解问题的过程，我只是把求解的结果写出来

the solution is that RK plus 1 is alpha times beta R K so RK is a point on the line RK plus 1 is another point on a line a multiple of R K by alpha beta alpha is the discount factor less than and beta depends on the xi okay

这个方案是r\_{k+1}等于alpha乘以deta 乘以r\_k，r\_k是线上的一个点，r\_{k+1}是线上的另一个点，alpha是小于1的折扣系数，beta是一个依赖于xi的函数值

this comes from this calculation this minimization

这个表达式是这个最小化问题算出来的

so I have this weight vector and notice if you look at this weight vector notice that's always greater than 1

我就有了这个权重向量，你注意这个权重向量，beta的值要大于1的

so alpha is some number less than 1 is multiplied by a number greater than 1 and the product could be greater or less than 1

所以alpha小于1，乘以一个大于1的数，结果可能大于1，可能小于1

if it's less than 1 then I have converters but if this beta is larger than 1 over alpha then we get divergence okay

所以如果他们的乘积小于1，这个算法就可以收敛，如果大于1，会发散

so for 1example okay so if alpha is greater than 1 over beta for example if this if you choose just the uniform norm okay xi is equal to 1 or maybe 1/2 to make it a problem distribution

举个例子，如果alpha乘以beta的值大于1，并且用均匀范数等于1或者1/2的分布xi

then you can see that this number here this number here becomes greater than 1 and you get divergence the method just diverges

然后你就可以看到，这个值大于1，算法就发散了

so the error bound that I gave you in the previous slide and does not hold because you don't have given for approximation

所以这时候上一页的误差就维持不住了，因为你并没有进行近似(成功的近似)

here's the zero reckon you just keep going off and there is no there's no this table is in the method fails

这时候算法就失败了

now it's a kind of a devastating example and stop provoking because it says that you just can't arbitrarily simulate project approximate and things will work out very easily you can get things not to work you have to be very careful about what you do notice that if you choose the xi properly

这是一个发散的例子，因为你不能用任意的仿真进行投影近似，这很容易失败，你必须非常小心地设计你的算法，正确地设计xi的分布

in particular if you choose xi two much larger than xi 1 then this becomes less than 1 and you get convergence

如果你选择了一个xi2远大于xi1的分布xi，他们的乘积会小于1，这时候算法会收敛

so the trick is to choose properly the distribution here of sampling

所以算法收敛的技巧是正确地设计采样分布

and more fundamentally the difficulty is that yes the T mapping is a contraction as we have seen in iterating with T convergence

一个更重要也更难得是映射T必须是压缩映射

however what we do here is we apply PI T and it's not necessarily true that the projection composed with T is a contraction

但是我们使用pi T的时候T就不必须是压缩映射了，这时候T组成的投影是压缩映射

NORM MISMATCH PROBLEM

who this happening here is what's called the norm mismatch problem

这一页就是我要讲的范数不匹配问题

and let me look at the picture again

我们再看一下这个图

remember that this fitted value duration start with some j 0 applies t to it projects down with respect to this projection norm again projects down and so on

还记得这个拟合值迭代么，从J0开始，使用T，然后范数投影到低维空间，再T，再投影，这么持续下去

so you generate a sequence of vectors J tilde 1 J tilde 2 J tilde 3 and so on by iterating with the mapping PI T

通过使用映射pi T迭代你就可以得到一个向量序列J tilde1，J tilde2，J tilde3…

and the problem is that Phi T may not be a projection may not be a contraction even though T is a contraction

问题就是即使T是压缩映射，pi T也可能不是压缩映射

if you didn't have intermediate with a projection there would be no problem but because of the intermediate projection you can get divergence because there is a norm mismatch

如果没有中间的投影，就不会出问题，但是有了中间投影，由于范数不匹配，算法就发散了

T is a contraction sum up sum norm but pi it's not the contraction projections are not contractions

如果T是压缩映射，pi不是压缩映射

but they are non expensive non expensive non expensive means that if they iterate on two vectors and you project those two vectors then the distance between the projection is no larger than the distance between the originals okay

这种现象叫做” non expensive”(不知道怎么翻译) non expensive指的是如果迭代的时候对两个向量投影，新向量的距离不会比原向量距离更大

I have two vectors projecting down the projections are closer together than the original okay

我有两个向量，通过投影到低维空间，他们应该比原向量距离更近

so if T is a contraction with respect to the weighted that norm this weighted norm then there is no problem

如果T是关于加权范数的压缩映射，是不会出问题的

T contracts vectors and projection does not pull them apart

T让向量靠近，投影不会让他们分开

so you still get a contraction and you get convergence

这样你可以得到一个压缩序列并且让算法收敛

however if you have T being a contraction with respect to one norm and the projection B with respect to another norm you may get a normed mismatch and create a divergence in the iteration

然而如果你有一个关于权重是1的范数的压缩映射T，B是关于其他范数的映射，这样就会产生范数不匹配问题导致算法发散

so what does this say it says that we need a special xi that works together with T in order to get a method that works

就是说我们需要一个特殊的xi和T一起作用让算法工作

just choosing an X I arbitrarily if the sampling distribution is arbitrary then you can get a real problem so that's the norm mismatch problem

如果随便选一个xi，再随便选一个采样分布，就会产生问题，也就是范数不匹配问题

finding a distribution X I such that projection with respect to that distribution composed with T is a contraction in particular find a magic way that normal under which T is a contraction and use that for sampling

找一个分布xi，让投影在这个xi下进行，同时T也是压缩映射，即使用xi采样，投影也在xi的作用下进行，就能避免这个问题

now it's a very central issue in approximation with the codex often and programming and simulation

这是近似动态规划和仿真方法中一个非常核心的问题

and we'll come back to this issue because it also appears in approximate policy direction

因为他还会出现在近似策略迭代中，所以我们还会讲这个问题

here we saw it we Michalek top approximated election

我们在这里看到的是它在近似值迭代中产生的问题

it also appears in the context of approximate policy direction

在近似策略迭代中它也会引起相应的问题

many questions can clarify this a little bit it's important to have the product of is to be a contraction and to choose Sai properly for this to happen

有什么问题么，正确地选择xi很重要，它可以让投影是压缩映射

APPROXIMATE POLICY ITERATION

okay so maybe we take a little break like like ten minutes just to digest this and then we'll go into approximate polish direction which will take a little longer to go to go through