Q-FACTORS I

okay we have been talking so far about costs of course function supported skillet optimum cost function now we are going to talk about I related quantity or the Q factor for the function prepare X cube compares for states and controls

之前我一直在谈论成本函数，现在我要说一下Q函数，与成本函数不同的是，它是一个状态与控制决定的函数

okay talk about you to practice our functional cares of state and control another state like jh state and control and by definition the optimal to partner of Xu is this expression here we explore next you asking you then just a well-defined one feet right and it is the first over of starting at x applying cube in the first stage

I'm gonna put it here，okay start again

我要把话筒夹在这，好了，我再讲一次

okay the Q factor of a pair xu is what you see here in this line by definition j star is the optimal cost function x bar is the next state so this is starting at state k the the cost of the first stage by using you and then plus you add the cost of all the remaining stages using an optimal policy so j star had x u start at x and u use an optimal policy all the way through q star of x in you you first use you some you whatever it is okay not necessarily optimal and then you use an optimal policy after that okay

你可以在这里看到状态动作对的Q值的定义，J\*是最优成本函数，x bar是下一个阶段的状态，所以Q值表示的是总成本，从时间k状态x开始，执行控制u的即时成本加上接下来所有阶段的最优策略获得的总成本，需要注意一点，J\*是后续阶段一直使用最优策略计算得到的，Q\*（x，u）的值是状态x时采用控制u，然后接下来的阶段使用最优的u，当前的u不一定是最好的，但是后续的u一定要是最优的

now this seems a little strange but if you look at this expression here it is the right hand side of Bellman's equation and you can write balanced equation in terms of q factors like this and also if i give you the optimal q factors you can calculate the optimal policy by minimizing this expression here rather than minimizing this entire expression

看这个有一点奇怪的表达式，这个表达式是bellman方程，的等号右边是Q值，如果你知道了最优的Q值，那么你就可以通过计算这个最小化问题获得最优策略而不用最小化原来的那个复杂的表达式了

okay now not only that but i can write the value iteration method in terms of two factors it's given by this expression here okay given qk the minimum of qk is is is the internet of via turret of value iteration and then QK plus 1 is given by this expression and if you minimize over you if I write the minimum of this over you you get exactly value iteration so it's another way of writing value iteration only this memory wasteful perhaps because now you maintain a function of both X and u however there's a big benefit from that because if you can calculate Q of X and u then you can obtain the optimal policy by means of a simpler minimization that's one idea behind you factories

不仅如此，我可以使用值迭代来计算Q函数的值，给定这个表达式之后，如果我知道了Q\_k的值，最小化这个表达式，就可以算出Q\_{k+1}的值，这就是精确值迭代算法，这种方法对存储空间消耗非常大，因为你现在需要维护一个状态动作对函数，他们都需要存在内存中，但是这种方法有一个很大的优势，就是如果你已经计算获得了所有的Q值，你可以计算一个很简单的最小化问题来计算最优策略

Q-FACTORS II

okay now to factors there are other ways to interpret them you can view them as cost functions for a different problem that involves an Augmented state where the states are x and u rather than just X so you have a pair X u and then we and then if you set up a bellman equation that involves this mapping here then that's bellman equation for this Augmented problem so in particular by using this interpretation all the results that we have for cost functions carry over to Q factors so you have value iteration for Q factors that is the same as value iteration for the four course you also have policy narration for Q factors and you have also a bellman equation where the mapping is not T but rather it is this F that operates on two factors to produce other Q factors

有两种其他方式·可以解释它，你可以把这个成本函数看作一个新的问题，这个新问题的状态是原问题的状态和控制，这样你就可以使用bellman方程表示Q函数了。特别地，你还可以直接使用值迭代算法与策略迭代算法求解最优策略，这时动态规划映射就不是T了，而是用F来表示。

now if you're doing exact if you are applying this all going to value it in policy direction exactly then mathematically whether you use two factors or costs is the same thing it's exactly the same mathematical operations you just keep track of a larger vector or a larger function so you need more storage but the amount of computation is equal and the results are mathematically the same however differences start properly occur when you are using approximations you use approximate value direction or approximate or Z Direction

现在你是用Q函数来进行精确的策略迭代与值迭代，不管是使用Q值还是使用成本，在数学上都是一样的，不管是要进行的操作，还是获得的结果，或者是计算量，不一样的只有Q函数你需要维护的向量函数更多，需要更多的存储空间。但是在哦我们开始使用近似方法（近似值迭代与近似策略迭代）求解时，他们就开始有区别了。

the main thing is that if you have optimal Q factors then you can obtain an optimal policy by this minimization and in the context of approximation if you obtain good approximations of two factors then you can get also an approximate policy through this equation and the nice thing is that you don't need to know G and the expectation operation also is not needed in other words you can compute policies in a modern free fashion okay

主要的是，如果你已经有最优Q函数的值，那么你可以通过最小化一个简单的问题获得最优策略，如果你有一个近似的Q函数的值，那么你可以获得一个近似策略，比较好的事情就是你不再需要知道g和这个表达式的期望，换句话说吗，你可以使用一种不依赖于模型的方法计算策略

let me explain this a little bit more in in in in the standard way of obtaining ten optimal policies is you compute J star and then you minimize over you this expression to do this minimization you need to have a model for G you need to have a model for the random variable involved here so you take the expected value you need to have a model for the system f to plug in here so given J star you need to have a model to compute an optimal policy however if you have Q star somehow then you don't need a model anymore you can simply minimize this expression here and that's that now this turns out to be a great thing approximate dynamic programming in many contexts will you have a simulator of a system but you don't know the internal dynamics and you have data coming in from the simulator and data coming out you use this data to calculate few factors and after you have the cue factors then you have a list of all these cues in whatever state occurs you look in your list and you obtain the optimal policy without having to know the model in any at any point

这个问题我要解释得更详细一点，标准的方式计算最优策略的时候，你需要计算J\*，然后最小化这个表达式，你需要知道g的表达式，还需要知道随机变量w的信息，然后你才能得到成本的期望值，你还需要系统的转移函数，才能知道x执行u后新的x是什么然后才能使用这个信息计算最优策略。所以给定J\*后，你还需要知道模型的信息才能计算最优策略，如果你知道了Q\*，就不需要模型了，可以直接用一种很简单的方式计算最优策略，这是一种很好的方法，在使用近似动态规划的时候，很多情况下你会有一个仿真器，你不知道动态激励，但是你可以从仿真器中获得数据，然后你可以使用这个数据计算Q值，计算结束后你就有了一个Q函数得列表，在一个状态产生的时候，可以从这个列表中查最优策略，这种方法完全不知道模型的情况下也可以获得最优策略

of course so far I have not told you how to calculate you factors other than by using the model however there's a method called q-learning that we're going to discuss later next week whereby we use samples of this expectation obtained by some black box some simulator in order to calculate the sort of approximate is mapping and to calculate by simulation these -star factors or some approximation to them

到目前为止，我还没有告诉你怎么计算Q函数的值，这种方法被叫做q-learning，我们会在下周讨论他，使用仿真器进行采样，然后计算近似映射对应的Q\*的值或者Q\*的近似值

and in the end no model is necessary two factors can be calculated in a model free fashion by sampling it turns out and that's the big thing about them however we haven't talked about approximations yet so we will postpone this discussion for next week but I just wanted you to know that there are these few factors and in exact when you're doing exact calculations they are no different than costs it's only in the presence of approximations that they make a difference

Q函数不需要模型，通过采样就可以计算法的值， 我们还没有谈到如何近似它，我们会在下周讨论他，我现在提到这个内容只是想让你知道，在计算精确结果时，他们没有任何区别，只有在对他们进行近似时才有区别

OTHER DP MODELS

okay now we have focused on discounted problems these are the easiest problems the ones for which this the strongest theory however there are other dynamic programming modules and let me give you some examples we're not going to cover them in any detail but just don't want you to be aware

现在我们关注折扣问题，这是最简单的问题了，有很强的理论支撑，解决折扣问题还有很多其他方法，我要给你们举几个例子，当然我不会深入讲他们的细节，我只是想告诉你们还有这些方法

first of all there are undiscounted problems where the discount factor is exactly equal to one there is no discount factor

第一种问题是无折扣问题，就是折扣因子等于1，也就是没有折扣因子

now how do you keep the cost the total cost of such a system finite

那么如何保证这个问题的总成本有限呢

well many of these problems involve either convergence or reaching exactly some kind of a cost free state well the cost does not accumulate any more

如果问题最终会收敛或到达一个没有成本的状态，这样成本就不会继续积累了，也就能够保证总成本有界

so for example there may be a special termination state some destination which once you reach then you stay there with no further cost like reaching a goal okay in minimum with minimum cost or a robot completing an operation at some time in the future and then after completes the operation there is no more cost associated with that

比如问题会到达一个特殊的终止状态，这些终止状态是确定存在并且已知的，一旦系统到达了这些状态，系统就不会再转移到新的状态，也不会再积累成本，你就可以最小化这个总成本了，或者一个机器人在将来的某一个时间执行一个操作，执行完操作之后就不会再产生成本了

now such a problem is called usually a shortest path problem or if there is uncertainty about this a Markov chain involved and some kind of stochastic and certainty in the system it means called a stochastic shortest path problem

这些问题通常被叫做最短路径问题，如果马尔科夫链有不确定因素，比如随机性，又兼做随机最短路径问题

there are also other problems where the termination stage is sort of reached asymptotically and of course accumulate to something finite we call this generally and discounted problems and they have their own theory it is not quite as strong as the for discounted problems but depending on the assumptions that you have you may get pretty strong results

还有其他问题，终止状态是渐近到达的，积累的总成本也会是一个有界的数值，我们把这种问题叫做一般的折扣问题，这些问题没有折扣问题那么强的理论支撑，但是在满足某些假设的时候，你还是可以获得一些比较强的结果的

another major type of problem is when time is not discrete you may have a finite number of states a Markov chain but it may evolve in continuous time like for example queueing systems queueing systems are typically finite or maybe countable number of states corresponding to the different levels of loading of the queue different numbers of customers in the queue and then a customer departs and the state goes down by one what a customer arrives in the state goes up by one however the time that it takes for the customer to depart may be random okay and may dependence on your control action and the time that it takes for a customer to arrive is also random our customers may be arriving according to a Poisson process they may arrive very quickly or may take a long time depending on the roll of the dice okay

另一个重要类型是连续时间有限状态数量的马尔科夫链，比如排队系统，排队系统是一个典型的状态数量有限/可数，但是状态的数量与队列的客户数量相关，如果一个客户离开，状态减少一，如果一个客户到达，状态增加一，客户离开的时刻是随机的，依赖于你的控制，客户到达的时刻也是随机的，客户可能以泊松分布到达，也可能到的非常快，还有可能很长时间才到一个，到达的情况完全取决于随机的信息(原话是骰子投出来的结果)

now if you have a transition that takes a long time and you use discounting continues time discounting the discount factor is going to be stronger for the cost associated okay

如果你有一个进行长时间状态转移的问题，你使用折扣连续时间问题来求解，这时成本就会与折扣因子强相关

if a transition takes a long time then the cost that will incur at the end of the transition will be discounted more strongly

如果一个系统长时间地进行状态转移，那么后出现的成本就会非常严重地被削弱

so what you have is essentially discount factors that are random and depend on the state and possibly on the control that you apply

所以一般情况下，折扣因子应该是一个基于状态与控制的随机变量

other than that these problems finite state continuous time problems which are also called semi Markov problems otherwise are pretty similar to discount the problems the main difference is this they can be viewed as discount problems with state in control dependent discount factors

另外，这些有限状态连续时间问题又被称为半马尔科夫问题，半马尔科夫问题与折扣问题很像，主要的区别就是半马尔科夫问题的状态与控制是依赖于折扣因子的

so what's going to happen is the T mapping is going to change somewhat to incorporate this this discount factor that's state dependent the T new mapping is going to change but everything else will say pretty much the same in terms mathematically speaking and these problems have about a strong theory are discounted problems

所以与折扣问题相比，半马尔科夫问题的变化就是T算子会改变一些，从没有折扣因子到有折扣因子，还是一个依赖于状态的映射，同时T\_mu也会改变但是除了这个，从数学角度上讲，其他的东西都没有变，还是有很强的理论性，这些有强理论性的，就是折扣问题

now let's look at another class of problems that is harder okay more difficult continues time and continued space models

现在我们来看一个更困难的问题，连续时间和连续空间的模型

now these are the classical automation problems that you encounter in practice like controlling an aircraft controlling a car controlling a robot controlling some kind of a process a chemical process the traditional control theory problems are typically continuous pace and often continuous time you apply control not are discrete intervals but that contain over continues over time

这是一类典型的控制问题，你在实际解决问题时遇到的控制一个飞机，控制一辆车，控制一个机器人，控制一些过程，比如化学过程，传统的控制理论关注连续空间和连续时间的问题，他们不是离散的而是连续的

now here there are some substantial difference from the screen time

屏幕上是这些问题的区别

first of all mathematically these are far more complex particularly if you have stochastic uncertainty there's some pretty substantial mathematics associated with with with the continuous time stochastic processes involved here that diffusion models they are models involved in continuous time white noise which are little calculus we are talking about some pretty heavy-duty mathematics

首先，从数学上讲，这类问题更复杂，特别是有随机或不确定因素的时候。这类问题有大量与连续时间随机过程相关的数学问题需要研究，还有白噪声，都会带来大量的数学上的工作

even for deterministic problems which are continuous time there are tricky issues associated with differential equations there is something called want reactance maximum principle that governs the the solution of is there's a is a very strong and theory that goes back hundreds of years or the calculus of variations the types of Euler and LaGrange they were considering this type of problems

即使是确定性的连续时间问题也存在与微分方程相关的棘手问题，比如“called want reactance maximum principle”(不会翻译。。。)，想要解决他们，需要有很强的理论支撑，百年前欧拉和拉格朗日也都在思考这一类型的问题

it is possible to address problems like that by dynamic programming based on time discretization mathematics here a little shaky and you need to be careful how do you what kind of statements you make but I'd like to give you an idea of how dynamic programming applies to these problems

使用基于离散时间的动态规划来求解这类问题可能会遇到一些问题，所以你需要注意一下，下面我会告诉你怎么使用动态规划解决这类问题

CONTINUOUS-TIME MODELS

basically you have a continued time system involving a differential equation or a system of differential equations

现在你有一个连续时间系统的模型，你知道这个系统的微分方程

so X of T is a vector function of time and you have a differential equation on the right-hand side there is X the state of the system and the control that you apply at time T from some constraint set

x(t)是一个关于时间的向量函数，等号右边是一个微分方程，x是系统状态，u是你在时间t执行的满足约束的控制

there's also a cost function that is an integral from 0 to infinity and typically for this interval to make sense you have to have some kind G has to be 0 at some state which we are trying to to regulate the system to

下面这行是成本函数，这是一个从0到无穷的积分，无限期问题一般都是这种形式的，同时我们要控制的系统中的一部分状态对应的成本必须是0

okay let's call J starts of X the optimal cost starting from initial state X

我们把从初始状态x开始的系统最优成本叫做J(x)

and let's try to address this problem by time discretization let's discretize the time in intervals of delta small delta then the system becomes discrete like this okay

我们把这个问题使用delta离散化，这个系统就变成了离散系统

we differentiate this derivative here XK plus 1 minus XK by PI Delta and we sort of assume that over the small Delta interval the control that you apply is constant so piecewise constant control but changing very frequently now you get now a discrete problem where the cost per stage is this G multiplied with Delta

离散后，x\_k 减 x\_{k+1}是delta相关的函数，使用delta将时间切片，在分段约束内进行分段控制，这个段分的比较小，需要频繁地改变控制，现在你就有了一个离散问题，每个阶段的成本是g乘以delta

okay so bellman equation for the Delta discretized problem looks like this

这种delta离散问题的bellman方程就像这样

okay I indexed by Delta and the optimal cost is the minimum over U of the cost per stage within this Delta interval plus the optimal cost at the next state which differs from the current state by this Delta times F okay

最优成本是在所有可行的u中找一个最小化当前成本乘以delta加上下一状态最优成本，下一状态是通过当前状态加delta乘以函数f获得的

now let's suppose that that we can take the limit as Delta go to 0 and we obtain the optimal cost suppose that this limiting operation is ok and it is okay under some conditions but you have to believe it about how you about these conditions

我们让delta趋于0，就可以获得最优成本，这种极限是可以执行的操作，但是你必须在一些约束条件下进行控制

then how does this Batman equation become in the limit take this

bellman方程在delta趋于0时可以写成这个形式

on the other side this is a constant not depending on u and divide by Delta and take the limit as Delta goes to 0 then Delta is going to be canceled out you get 0 on this side and on the other side you get the inner product of the gradient matrix of J star gradient with respect to x times f of X you okay

这段没明白他想怎么解释，但是译者的理解是，导数等于0的时候能得到最优值

this is called the hamilton-jacobi bellman equation Hamilton Jacob you lived in the in the 18th century okay and so this equation dates from way back bellman is more contemporary and and he played around with this equation quite a bit and brought it to life again in the 20th century

这就是HJB方程，HJB方程出现在18世纪，用的比较少，bellman的研究让它在20世纪又被人重视起来

so his name is associated with it it's a famous equation not to be confused with a discrete-time bellman equation this is the continuous-time analogue

所以他的名字就与这个著名的方程联系起来了，这个方程描述的不是离散时间问题，而是连续时间问题的方程

and it involves the gradient here of J star how do we know that J star has a gradient well the answer is we don't know we hoped it has a gradient but we don't really know

这个方程包含了J\*的梯度，但是我们怎么确定J\*有梯度呢，答案是没法确定，我们希望它有梯度但是不知道他是不是真的有

they have two big conditions to guarantee that it's quite possible that J star has some corners but may even be discontinuous some points ok

有两个条件能保证J\*有梯度，比如J\*存在一些corner(不知道怎么解释，但是感觉是拐点)

but assuming that this equation this discrete is this limiting operation is correct you have the situation and now the optimal policy can be obtained by minimizing for each X in here like in balanced equation

假设这个离散与取极限的操作是正确的，就可以通过对所有的x最小化右边的表达式获得最优策略了

now notice an interesting thing okay let's so let's see how policy direction will work informally policy direction for continuous-time systems goes as follows given the current policy do a policy evaluation which is to find a function J mu that satisfies this equation without the minimum here of course this equation is obtained from the bellman equation for the new policy by taking the limit as Delta to go to zero then do a policy improvement which is to minimize this expression with J mu and get a new policy nu bar that's a legitimate policy direction Albert that people have been using increasingly in the last few years

现在我们来看一个很有意思的东西，就是连续时间问题中策略迭代是怎么工作的。给定一个策略mu，使用ballman方程对策略mu进行策略评估，不是最小化，而是直接计算这个方程的解J\_mu，然后在delta趋于0的时候最小化这个bellman方程进行策略改善，得到新的策略mu bar，这是一个人们这些年一直在使用的方法

however there's a very interesting thing to not be noted in order to do policy improvement we need to know the gradient of J mu not J mu it's the gradient that we need to know and this perhaps should give us some pause to think about the discrete-time case should be we be trying to learn cost values or cost difference values okay we'll come back to this point again in the next lecture but for the moment I'd like to notice that as time it becomes continues than the character of this equation the bellman equation changes at this particular point

但是有一个很重要的事情要注意，就是为了进行策略改善，我们必须知道J\_mu的梯度，不是值，是梯度，这与离散时间问题是不一样的，我们必须去学习成本值与成本的问分值，我们会在下一次可将这个问题，我现在只是想要告诉你，连续时间问题与离散时间问题在这个地方是不一样的

okay have any questions so we have all these many problems perhaps we'll touch upon some of them at some point but in but really there are many different variations of problems and we would like to have a unified way to treat them we don't want to prove the same theorems over and over again for slightly different versions of the same problem so what I'm going to get into now is a couple of more advanced topics that have to do with a unified way of dealing with dynamic programming problems

我们要解决的问题种类很多，我不想每次解决一个问题的时候都对这些定理都证明一次，因此我希望能有一个统一的方式来处理他们，所以现在我要进入一个更高级的部分，用一个统一的方式解决动态规划问题

A MORE GENERAL/ABSTRACT VIEW OF DP

so let's let's look at a more abstract view of dynamic programming and just to just as a mathematical preliminaries suppose that you have a space any kind of vector space with some norm RN could be one a function spaces could be another and a contraction mapping is a mapping f that map's Y in the space into itself and has the property that that when you apply this F to two vectors Y & Z you get their distance to become smaller than the distance of Y & Z so this is the definition classical definition of contraction mapping Rho is something between zero and one it's called the modulus of contraction of F that's definition

让我们来看一下更抽象的动态规划，假设你有一个向量空间R^n，任意形式的向量空间都可以，这可以是一个函数空间，使用一个映射F，从Y映射到Y，这个映射F有这样的性质：你对y和z使用算子F，得到的新函数的距离小于y和z的距离乘以rho，rho在0-1之间取值，之前讲过这个性质，叫做压缩映射，这就是一般性的动态规划的定义

in an important example that's relevant for us is that if you let X to be a set like the state space in dynamic programming and V is some positive value function a functional f of X that is takes positive values let's consider B sub X which is the set of all functions on X such that the ratio of J and V is bounded over X J itself may not be bounded but when scaled with V it becomes bounded okay if V is identically equal to one then this is just the set of all bounded functions

有一个比较重要的例子，如果你定义动态规划的状态空间X，正值函数v(x)，定义B(X)是X上的所有函数J的集合，J比v的值就被x限制住了，J本身没有被x限制，当引入v的时候他们的比值就被限制了，如果v的值一直等于一，那么那么这就是一个有界函数的集合

and we can define a norm on this function space the weighted soup norm which is the maximum of this ratio here so if V is identically equal to one you get the case of the maximum norm but this is a scaled norm and I'll show you why this is convenient this is the important special gift for the discount problem the mapping is T and T mu for V identically equal to 1 and Rho equal to the discount factor they are contraction mappings in the sensor

现在我们定义一个函数空间上的范数：sup-norm，它是J的绝对值除以v的最大值，如果v的值是一，你就可以得到最大范数，但这是一个标量范数，下面我告诉你为什么说这个定义很好用，如果一个折扣问题，映射是T和T\_mu，v值等于1，rho的值取折扣因此，这就是一个压缩映射了

CONTRACTION MAPPINGS: AN EXAMPLE

ok let me give you an example why this contraction mapping ideas may be useful in dynamic programming suppose that we want to we have a problem where the state space is not finite but rather it is countable let's say the states are 1 2 all the way to infinity like for example a queueing system with infinite storage you can have 0 customers one customer to any number so a number of states is countable and not only that but the cost per stage which depends on the state may not be bounded for example if you have a hundred customers there will be a greater cost than having Tadcaster less than if you have an infinite number of customers you may want to have an infinite cost so there may not be a bound on the one stage cost function naturally in some of these problems however if we introduce a weighted norm which takes into account the change in the cost function then the cost per stage will become bounded but in the sense of the weighted soup normal ok

我举个例子说明为什么压缩映射对于动态规划是有用的，加入我们有一个无限状态集合，但是这个集合状态数量可计量，也就是说状态编号为1，2直到无穷。比如一个存储能力无穷大的排队系统，可能有一个客户，两个客户或者任意数量个客户，也就是说状态的数量是可数的，平均每阶段的成本依赖于系统状态但是可能没有上界，举个例子，如果系统中有100个客户，那么当客户数量大于100的时候，成本也可能超过100，你希望一些问题中每阶段成本都有上界，我要向你介绍一种加权范数（weight norm）来把成本函数变得有上界

now here is a theorem that says that if you have the mapping T\_mu to have this generic form so remember i is here the the states one two and so on and let's say the T\_mu mapping has the form a constant which represents the cost per stage a state I and also it is linear with coefficients AI J being being like the transition probabilities for example from I to J suppose that you have a T\_mu has this form where bi and a\_{ij} SON scalars then T\_mu use a contraction mapping of modules row if and only if this relationship holds now if the a IJ is our transition probabilities they add up to 1 but when scaled with this V that schemes also the one stage cost if if you have this relationship holding in other words the future V okay a\_{ij} depends on J so V may be increasing relative to Y but if a\_{ij} decreases faster as J increases then you make up this relationship holding and then you may have a contraction mapping for team you

有一个定理，有一个通用形式的映射T\_mu，i是系统状态，T\_mu映射是一个常数，表示每个阶段的成本，这里面的成本是一个线性函数，系数a\_{ij}可以是状态i到状态j的转移概率，假设你有一个这种形式的映射T\_mu，包括标量bi和a\_{ij}。当这个表达式成立的时候，T\_mu是一个压缩映射，如果a\_{ij}是系统的转移概率，累加等于1，v是每个阶段的成本。a\_{ij}依赖于J，所以v的值增加与Y相关，同时a\_{ij}减少的速度比J增加的速度更快

and if you have a contraction mapping for TM u then the corresponding T mapping is also a contraction mapping

如果你有一个压缩映射T\_mu，那么与他相关的映射T也一定是一个压缩映射

so if your model satisfies this condition which in queueing systems is sort of a standard condition then you have all the necessary structure to obtain the strongest possible results bellman equation has a unique solution there's an optimality condition there is a value iteration works policy Direction works everything that works for finite state space and bound at one stage cause extends to unbounded one stage cost provided you can apply this kind of structure so this is an example where you can extend the range of the nice results by using some analysis involving contraction

因此，如果你的模型满足这个条件，在排队系统中这是一个标准条件，那么你就有了bellman方程必须的结构，有一个唯一的解，这也是最优条件，值迭代与策略迭代都能够求解有限状态空间并且每阶段有上界的问题，同时可以扩展到没有上界的问题，刚刚讲的内容就是一个很好的把有上界问题的方法扩展到无上界问题的例子

CONTRACTION MAPPING FIXED-POINT TH.

generally speaking with contraction mappings the main result is that they have a unique fixed point this is a classical result from analysis if you have a contraction mapping F with modulus Rho then it has a unique fixed point within the set of B sub X of bounded functions so in the context of dynamic programming this is just translation to the fact that Bellman's equation has a unique solution furthermore if J is any function B sub X then if you apply F repeatedly to J then in the limit you get J star and you have also this estimate of the error okay the error of F case of J - J star decreases with Rho to the K as as K increases

一般来说，压缩映射的主要性质是有一个唯一的不动点，这是要给经典的分析结果，如果你有一个模型rho下的压缩映射F，这个模型在有上界函数集合B(X)中就有一个唯一的不动点，在B(X)中的任意函数J，多次对J使用映射F，如果使用很多次，你可以得到J\*，同时可以估算这个J\*的误差，J-J\*的误差会随着k的增加而减小

since the classical contraction mapping theorem and and okay this is the proof of this you can find in the textbook and there is also there okay there's also group all the mathematical details are there that's the that's the key the main theorem for contraction mappings now what we would like to do is leverage the power of contraction mappings and into dynamic programming so that we don't have to deal with all these different models that that that very insignificant details we don't want to deal them with them separately

由于有经典的映射定理理论，他们的证明的一些数学上的细节可以在教材上找到，，我们想做的是使用压缩映射处理动态规划问题，这样我们就不需要处理不同问题的各种细节，而是可以使用一个统一的理论完成

ABSTRACT FORMS OF DP

so let's look at an abstract form of dynamic programming based on the fundamental properties of monotonous T and contraction

我们来看一看基于单调性和压缩性的抽象形式的动态规划

suppose R sub X is the set of real valued functions J and let's introduce a mapping H that takes States controls and functions J into other functions T sub J so instead of dealing with expected G Plus alpha J so on we abstract it more generally

假设R(X)是实函数J的集合，我要介绍一个映射H，从X，U和J到函数TJ的映射，这样我们就可以处理H，而不是处理更复杂的g加alpha乘以J的期望。

and let's introduce policies in a more abstract way let's consider the set of policies which are functions mu such that muse of X belongs to some constraint set for every state and consider the mapping T\_mu associated with a policy mu which is obtained from H when you plug in mu let's consider the problem of finding a solution to this equation J star equal to TJ star so this is very similar to what we have been doing except we use a more abstract function H for the expression in the dynamic programming algorithm now if we introduce this framework we simultaneously can deal with many many different problems

让我们用一个更抽象的形式来介绍策略，考虑一个策略mu构成的集合，mu(x)对于任意x都满足一些约束，考虑从H观察到的策略mu的映射T\_mu，现在要找满足J\*=TJ\*的方案，这和我们之前提到的优化问题是相似的，但是我们用了一种更抽象的函数H来表示动态规划算法中的符号，我们介绍了这些东西，以后就可以用这个理论解决很多不同的问题

EXAMPLES

one example is the discounted problems that we have been looking at which is this expression here one stage cost plus alpha times the J function

给一个折扣问题的例子，看这个表达式，一个阶段的成本加上alpha乘以J

another case is the semi markov problems I was talking about earlier the continuous-time problems after all the mathematics are said and done then it turns out that the right expression for V that you should be using is the sum over one stage cost plus another expression here which involves this this state dependent discount factors I talked about earlier so instead of having here alpha times the transition probabilities you have some numbers that involve I could that in that add up to something less than one and also but are have a little different section than here

另一个例子是半马尔科夫问题，我们之前讨论过的连续时间问题，树须田表达式是这样的，等号右边是当前阶段的成本加上一个状态相关的折扣成本，但是这个折扣不是alpha而是转移概率，这个概率小于一，与上面那个表达式有一点不一样

a third type of example arises in game theory sequential games where W is not chosen randomly not random but it's chosen by some opponent so I choose my u now in my opponent looks at what I chosen and chooses W from sunset trying to make things bad for me then I choose another you after having looked at what he has done then he chooses another W and so on this again has a dynamic programming structure pretty similar to the Socastee case from one state I apply a control then some W occurs and then I go to a new state apply another control and so on it's just that W is not a random variable but it's chosen antagonistically so instead of having an expected value you have a maximization and the corresponding mapping becomes like this so minimax control problems robust control worst-case control games against nature game suggests antagonistic opponent all fall into this category

第三个例子是在回合制游戏理论中的应用，w不是随机选择而是从对手的行为中选择，现在我选择了一个控制，我的对手观察到了我的选择然后选择了一个w让局面变得对我更坏，然后我选择了一个新的w，对手又选择了一个w，就这样进行下去知道游戏结束，这是一个很完美的随机动态规划的结构，在某个状态我选择了一个控制，一些随机因素产生作用，然后观察到新的系统状态，执行另一个控制，就这样持续下去，w不是一个随机变量，而是对抗性的选择，不需要用期望来评价，你使用相关的映射做一次自己利益的最大化，就相当于做一次鲁棒控制，在最坏的情况下做出最利于自己的选择，有对手的对抗性游戏都属于这一类问题

here's another example shortest path problems okay it's a very simple type of problem you have a graph with n nodes in the destination node each time you go from one node to another node there's a length associated with it or cost and you start at any one node and you want to find a path that has minimum length a path to the destination that has minimum length you can view this as a dynamic programming problem where the states are X okay the nodes of the graph U is the choice of next node that you make the branch that you go from that I love which you go from the current state and J is is a cost function associated with the different nodes in the graph so the H mapping has this form except if if if d if u takes you directly to the destination in which case there's a zero cost after that so this is the deterministic shortest path problem there's a stochastic version of this problem where when I choose you the next node is chosen according to a probability distribution which represents W or there is a minimax version of this problem where the next node is chosen by an antagonistic opponent in which case I get a none discounted version of this type of thing here all of these problems can be cast as special cases of the general framework and I can use a unified theory to treat all of them simultaneously these and a few more ok

这是一个最短路径的例子，这是一个非常简单的问题，你有一个n个节点的图，知道目标节点，每一次从一个节点到另一个节点都会产生相应的长度或者成本，你从任意一个节点开始，想要找到一个路径让从起点到终点的路径最短，你可以把它看作一个从状态x开始的动态规划问题，图的节点是你在当前节点选择的控制u，然后就可以从当前节点到选择的节点u。J是图上不同节点开始走的成本函数，所以映射H是从当前节点直到终端节点的映射，需要注意的是终端节点的成本是零，这就是确定性最短路径问题。如果你选择了一个节点，但是新的节点是根据概率分布跳转产生的，表现为w，这就是一个随机最短路径问题，如果从minimax角度出发获考虑这个问题，即下一个节点是被对手选择的，不使用折扣因子。这就是一个很通用的框架，我可以使用这个统一的理论处理所有相似的问题

ASSUMPTIONS

so the assumptions I'm going to use here are first of all monotonicity of this mapping H

我要给出的第一个假设关于映射H的单调性

if you have two functions J and J Prime one of them higher larger than the other then applying h maintains the the the direction of the inequality

如果你有两个函数J和J’，已知一个大于另一个，对他们使用映射H可以知道映射后的函数值大小关系不变

and we can show all the standard analytical and computational results of computational dynamic programming within this more general context if monotonicity is satisfied

如果满足单调性，我们可以在一般的问题中使用所有的动态规划的标准分析与计算结果

and in addition you have a contraction property and the contraction property for this mapping h is that a map in T mu and TJ are contractions

另外如果模型具有压缩性，即压缩映射H具有压缩性，那么T\_mu和TJ也具有压缩性

for every j T\_mu of j and TJ belong to B sub X and for some alpha and all mu and J prime J you have this contraction property

对于每一个J，T\_mu J和TJ在集合B（X）中，对于一些alpha和所有的mu，J’和J都有压缩性

so the contraction assumption holds if the T\_mu mapping which is obtained from H by plugging new in terms of instead of u has is a contraction

假设压缩性成立，把H中的T\_mu代替控制u，得到的结果还是具有压缩性

if you just have on the tonicity then as you program in undiscounted problems

如果你只是在写一个非折扣问题的程序

then you can still show various forms the basic results under some conditions but the fear is not quite as strong

你会遇到很多不同形式的问题，他们满足一部分条件，这个时候这些理论就不是很强了

and a weaker substitute for the contraction assumption is something called semi contractive

一个比较弱的压缩性的替代品被叫做半压缩性(semi-contractive)

nests who are roughly speaking for some policies mu T mu is a contraction but for others it is not

这个性质只对一部分策略mu和映射T\_mu成立，对其他的不成立

also there's some additional assumptions that guarantee that the non contractive you are not optimal

所以这些额外的假设无法保证压缩性，也就没法保证最优性

so essentially you're operating within the range of the contractions and and and the contractive Theory comes into play more or less

所以你必须想办法让问题具有一定程度的压缩性

we're not going to go into great detail of this I just want to sensitize you that there are these extensions of the theory and we're not just dealing with discounted problems

我不会讲太多细节，我只是想让你知道这些理论的扩展性质是什么，同样我不会关注非折扣问题

RESULTS USING CONTRACTION

and I'll just sketch the results

这是结果的草图

if you have this contraction and monotonicity then

如果一个模型同时具有单调性和收缩性

first of all T\_mu and T are weighted sup-norm contraction mappings with and they have unique fixed points so Bellman's equation holds based on the contraction properties of H

首先，T\_mu和T是加权范数压缩映射，我们可以得到唯一的不动点，这时候基于映射H的bellman方程的收缩性成立

the second is that value iteration converges within the small general setting starting with any J and applying T\_mu you get the cost corresponding to a policy similarly starting for T starting at any J if you apply this mapping T repeatedly in the limit you get J star

第二个性质是值迭代的收敛性，从任意一个J开始，不断地使用映射T\_mu，最终你可以得到J\*

you have that you have an optimality condition that that mu attains the minimum in the definition of T if and only if J mu is equal to J star and the proof is very similar to what we gave for discounted problems

你可以得到mu的最优条件，即J\_mu=J\*，证明很简单，下面是折扣问题的最优性证明

RESULTS USING MON. AND CONTRACTION

if you have assuming both one of the Nestene contraction you have that that the fixed point of T J star is the minimum over all J mu you also have that for there always exists for any epsilon and epsilon optimal policy a policy that attains the optimum within epsilon

如果你假设这是一个压缩映射，对于任意epsilon最小化J\_mu你就可以得到TJ\*的不动点，即epsilon最优策略，在epsilon下获得的最优策略

you can consider also non-stationary policies

你也可以考虑非平稳策略

and you may define for any non stationary policy the cost associated with it like so

你可以定一任何非平稳策略的成本

in pretty similar way as for the discounted case and finally that J star is not only a minimum of a stationary point is not only optimal over stationary policies is also an optimal over non stationary policy as well

与折扣问题非常相似，最后J\*不仅是平稳策略的最优策略，也是整个策略空间内的最优策略

if you recall I gave you a reference to a research monograph that I just finished about a year ago and all of this abstract dynamic programming stuff is part of that monograph so I couldn't resist the temptation of telling you about

如果你还记得关于我给你的去年写出来的那本参考书，抽象动态规划，我之前讲的所有的的单调性的研究都是那本书的一部分，所以我忍不住要告诉你，可以去看看这本书

THE TWO MAIN ALGORITHMS: VI AND PI

okay so now also we have algorithms of valuing policy duration with this this more general framework value iteration is is convergent in this sense here for any starting J policy direction is defined in the abstract way that we have seen earlier given a policy mu K we evaluate it like so by finding it as a fixed point of the map continue okay find UK plus one by taking the minimum in the in the T mapping and it's everything goes is defined in the usual way there's an optimal optimistic version of policy direction where the policy evaluation is done with a finite number of values directions rather than with an infinite number of value durations which is policy duration or with one which is value iteration and so everything goes through with in the abstract framework and we cover in this way an enormous range of problems without much analysis and with a common understanding for all of this so that's one theoretical offshoot that I wanted to sensitize you to and the second one has to do with our synchronous algorithms

现在我有值迭代和策略迭代，这是更通用的框架，值迭代在这里面是收敛的，对于任意的初始函数J，策略迭代被我们之前讲的抽象方法定义，策略迭代中评价mu\_k的值，可以找到映射的不动点，在映射T中通过最小化bellman方程进行策略改善获得mu\_{k+1}，这里面所有东西都是使用通用方法定义的，这是从最优化角度看待策略迭代的，在有限集中进行值迭代进行策略评价要好于在无限集中进行策略评价，这些东西在讲义中都是使用抽象符号描述的，我们用这种方式讲了一种问题的理论而且可以扩展到所有问题，这样就不用重复地推导和分析了，这是我想让你敏感的第一个问题，第二个问题就是我们接下来要讲的异步算法

Q&A

but let's take a break for ten minutes and then we'll get back to that okay have any questions yes I'm wondering if we don't know there are many examples wondering if we don't know the distribution about uncertainty how to model these problems with the worst case we don't amount of customer butter we can observe we can use the history informations first several stage we don't know the if the Japanese produced I even will however a long time we don't know the exactly introducer yes and we may update our update our estimation equality tribution but can we improve okay the question is the following we have okay we have this stochastic model and perhaps we know that stochastic but unfortunately we don't have a model for W we don't have the probability distribution so you can either view this there are many different ways to address the situation one possibility is you may not know the distribution but we may know some bounds on it in that case you may adopt a not a pessimistic viewpoint our worst case you point and model it in this set in in this set membership fashion okay you don't know the distribution you can take expectation but you know bounce so we take this worst-case viewpoint and then you fall into the minimax case that would be one way to do another way to do it is perhaps you have the distribution maybe unknown but you know that it has some form like for example it may be a Gaussian but you may not know the mean yes right a parametric type of method and then what you need to do as the system is operating you need to estimate at the same time the parameters now problems where you have what the model has unknown parameters are known as adaptive control problems okay and adaptive control problems are covered by this methodology however you have to view them in a more general setting to introduce special state value addition of state variables which model the unknown dynamics or the unknown parameters in the system finally there's another possibility that even though you may not know W you may have you may not know the distribution of W you don't have a closed form you don't know it's Poisson dance you know it's dancing whatever however you may have a simulator of W then you fall into the model free case that I talked about earlier and you may use the cue learning ideas so there are all these various possibilities but let me also add but if you don't know your model is generally a bad thing not a good thing there are a few things you can do however okay so let's take a break and be back in mins exactly