AI 算法进阶（Advanced AI System）

@Tyler

4. 行为主义：控制论与强化学习基础

目录

- **模块一：行为主义与控制论**
- 模块二：强化学习入门（MDP 与 Q-Learning）
- 模块三：深度强化学习（DQN）
- 模块四：从强化学习到语言智能体预测控制

从 类ChatGPT 服务背后的智能控制说起

- 在今天，每个用户都可以像与真人对话一样，通过 Web 页面与 ChatGPT 等大语言模型进行自然互动。
- 界面看起来只是一个简单的对话框，但你是否想过：
- **“同样一句话，你用的，其实并不一定是同一个模型。”**
- 甚至——
- **“不一定是同一种算力，甚至不一定是同一个服务策略。”**
- 你看到的：
- 用户：每天通过网页自然交互，仿佛一个模型在回应你的每一句话。
- 你没看到的：
- 系统：背后运行的是一个由多个模型组成、能自动切换策略的智能调度系统。

从 类ChatGPT 服务背后的智能控制说起

表面上，它是一个永远在线的聊天框；
实际上，它是一套**基于反馈控制思想构建的多模型动态服务系统**。

我们不再手动选择模型，而是构建了一套能根据实时状态自动决策的智能系统——

一个 **能感知使用成本、预判用户行为，并自主切换模型的控制中枢**。

接下来，让我们拆解这个系统的底层运行逻辑，看看它如何在“用户体验”与“平台成本”之间，做到动态平衡。

一个体验完美、成本失控的系统，注定不可持续

在用户看来，一切都很丝滑：

回复迅速，内容智能，对话毫无卡顿

不限次数、不限内容，月付 ¥99 即可畅聊

但在系统背后，我们正面临一场**看不见的运营失衡**：

现实中的使用差异：

用户类型	使用方式	带来的压力
♀ 日常用户	偶尔提问，使用频率低，计算资源占用少	资源闲置、浪费
重度用户 / 羊毛党	高频提问、批量操作、长上下文、强制大模型调用	成本激增、严重亏损

控制问题的本质：

用户付的是订阅费，是**固定收入**；
用户调用的是云端模型，成本却**高度波动**。

我们必须在不影响用户体验的前提下，**动态控制系统的“服务强度”**，让成本不至于失控。

一个体验完美、成本失控的系统，注定不可持续

- **简单封顶策略不够用：**
- 固定次数限制 → 影响体验、破坏产品心智
- 静态配额机制 → 无法适应用户行为变化
- 我们需要的，不是一套僵硬的规则，而是一个**能实时感知、灵活应对的智能控制系统**。

- 如何构建一个“**会自己调节的系统**”，在体验与成本之间找到动态的、可持续的平衡点？
- 我们将用经典的控制系统结构，来重新理解大模型服务系统的“操作中枢”。

OSI 指标：大模型服务的核心控制信号

极客时间 训练营

- **为什么需要一个指标?**
- 控模型开销、保用户体验，我们需要一个**量化指标**来判断：
  当前用户的使用情况是否“偏离了合理轨道”?
- **OSI：On-Schedule Indicator（进度占比指标）**
- OSI 计算公式：

$$\text{OSI} = \frac{\text{用户截止今日已产生费用}}{\text{总成本订阅费} \times \frac{d}{D}}$$

OSI 数值	状态判断	控制建议
OSI < 1	使用偏保守	可继续使用大模型
OSI ≈ 1	使用刚好	保持当前模型策略
OSI > 1	使用超前、超支	应降级模型，减缓成本上涨

OSI 指标：大模型服务的核心控制信号

极客时间训练营

- **OSI 的核心作用：**
- 作为**反馈控制系统的观测信号**
- 驱动模型选择的核心依据
- 实现“超前干预、自动降级”的基础前提

- **现实风险提示：**
- OSI 超过 1 并不是瞬时触发，而是一个延迟反馈后的“后果显现”
  下一页我们将解释为何“提前干预”至关重要

为何不能等 OSI 爆表再降级?

- **工程现实：统计延迟与切换滞后**
- **看似实时，其实存在系统延迟：**实际上，用户每次调用大模型消耗的token，需要**统计、结算、上传**，往往有几十秒到数分钟的“系统延迟”

阶段	描述
用户调用	请求发出，可能瞬间触发多个大模型调用
成本统计	Token 使用量上传、计费系统汇总
控制器接收	OSI 值刷新需要数秒到数分钟不等
策略执行	控制器做出反应并推送模型切换指令

- **风险场景举例：**
- 用户在某一时刻连续发起大量 QWen-32B 请求
  OSI 实际早已“爆表”，但控制器此时还没收到更新
  等你看到时，亏损已经发生

为何不能等 OSI 爆表再降级?

- **解决方案：引入提前干预机制**
- **安全缓冲区 + 预测机制**

控制策略	描述
提前触发阈值（如 OSI > 0.95）	还未超支时就预先执行降级，降低风险
趋势预测	利用历史用量趋势，预判未来几分钟内是否爆量
弹性窗口	控制系统拥有“踩刹车”的提前权，而非撞线式被动反应

- **控制系统从此目标转变：**
- 不再是“等问题发生再修正”
  而是“预测问题、主动规避”

本质上，这是一个“反馈控制系统”

当我们说「要让系统在体验与成本之间动态平衡」，其实就是在构建一个**智能闭环调节系统**。

控制论告诉我们：

凡是要“稳定一个目标值”，并“根据反馈做出调节”的系统，**本质都是控制系统**。

用控制系统的三要素重新建模：

元素	在大模型服务中的对应含义
参考值（Setpoint）	每个用户应占用的 **目标资源预算**（如月 ¥99 的换算成本）
反馈量（Process Variable）	用户当前实际产生的 **累计模型调用成本（如 OSI）**
控制器（Controller）	根据偏差，实时调整 **大模型使用概率 / 服务策略 / 降级行为**

闭环控制流程图：

智能控费机制：PID 控制器

系统特性决定了我们必须使用反馈控制：

用户行为 **不可预测**（非线性、突发性）

模型成本 **不可精确提前估计**

延迟存在，**一旦失控就很难补救**

这不是一次性计算能解决的问题，而是**需要持续感知、持续调节**的过程。

那我们用什么方法来构建这个智能控制器?

我们将介绍：最经典、最易落地的反馈控制器——**PID 控制器**，它如何成为构建大模型控费系统的第一代核心机制。

智能控费机制：PID 控制器

在工业控制、机器人、航空航天等领域，**PID 控制器**（比例–积分–微分）已被广泛用于各种自动调节系统。

在大模型服务系统中，PID 同样可以承担起“**自动控费调节器**”的角色。

控制目标回顾：

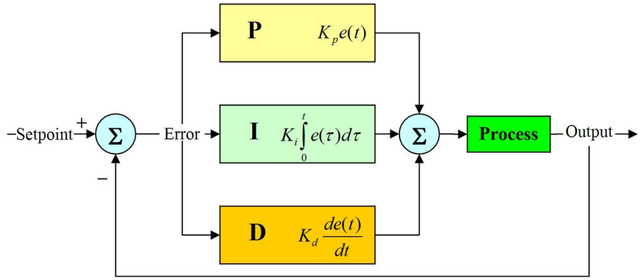
我们希望：

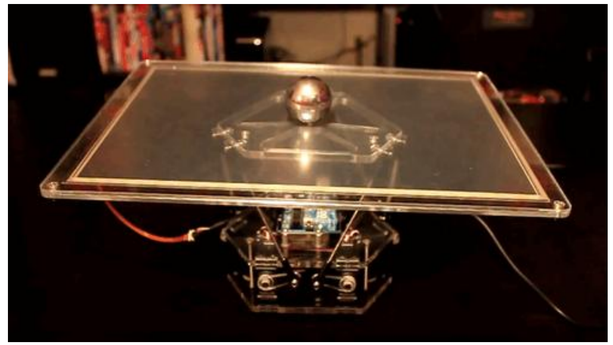
用户能持续使用大模型服务，**不影响体验。**

系统能实时感知成本趋势，**提前调节策略。**

PID 控制器：第一代智能控费调节器

- **什么是 PID?**
- PID：**比例–积分–微分控制器**
  是控制论中最常用的一种**反馈调节算法**
- **控制公式：**





$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

模块	含义	控制作用	智能意义
P	比例控制 Proportional	当前偏差越大，控制响应越强（快速调整）	误差过大
I	积分控制 Integral	考虑历史累计误差，消除长时间偏离（稳定性增强）	持续不消失
D	微分控制 Derivative	感知误差变化趋势，抑制突发波动（提前反应）	快速放大

- **在大模型控费中的作用**

符号	含义
u(t)	控制输出（决定调节幅度）
e(t)	当前误差：OSI目标值 - 实际OSI
Kp,Ki,Kd	三个参数：分别控制反应速度、累积偏差修正、抑制振荡

智能控费机制：PID 控制器

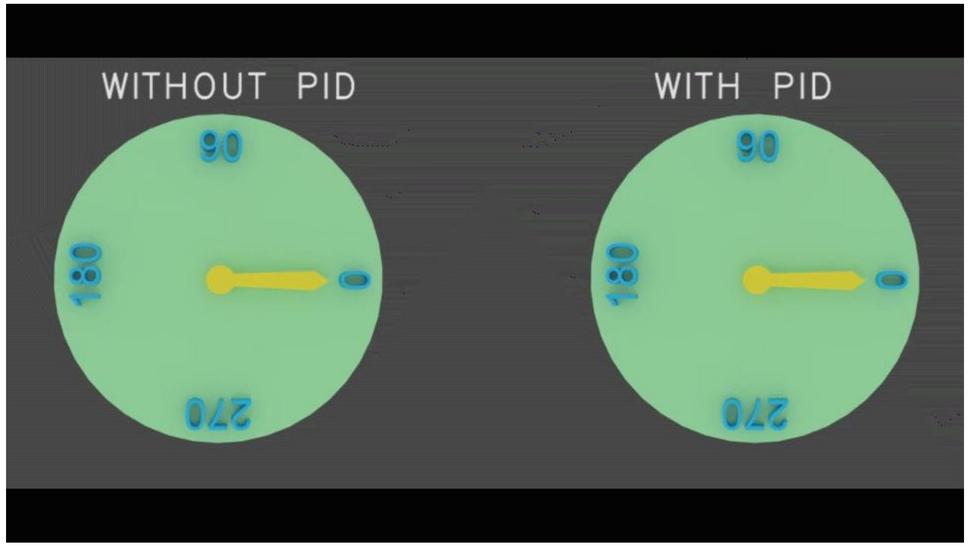
- **控制器输出 → 策略映射方式**
- PID 控制器的输出 u(t)∈[0,1] 表示当前用户使用成本与平台预期之间的**偏差强度**。我们可以将这个连续控制量映射为多种实际的调节策略，以在不损害体验的前提下，实现动态控费。

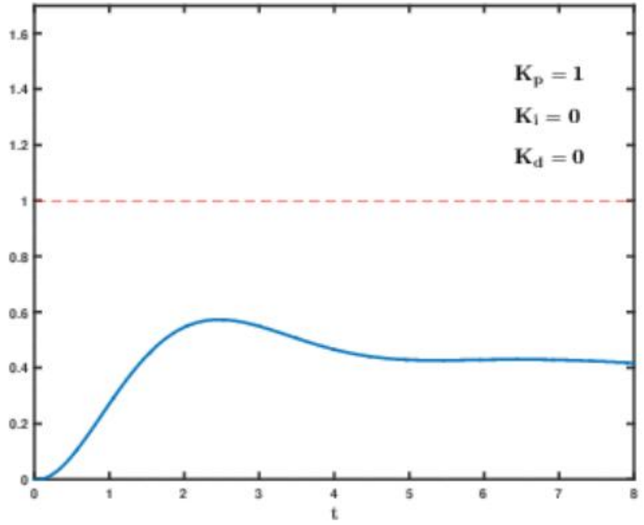
u(t) 值段	系统策略	用户感知
0.0≤u(t)<0.40	全量大模型放行	最佳体验
0.4≤u(t)<0.70	按比例放行，逐步倾向轻量模型	偶尔出现“轻量回复”
0.7≤u(t)<0.90	启动降级策略，切至 8B 模型或压缩服务	质量下降，可提示限流提醒
u(t)≥0.9	强制退回默认模型，关闭插件 / 禁用长上下文等功能	服务简化，提示配额用尽

- **工程优势：**
- 结构简单、响应快
  可用于构建**无人值守自动控费系统**
  是 AI 系统中最早的“自调节机制”落地实践
- 然而，这种被动调节方式也有明显局限……

PID控制器的局限性：为何我们还需要更智能的机制

- **依赖经验调参，难以泛化**
- 不同用户、时段、场景的控制需求差异大
- PID 参数（K_p, K_i, K_d）无法“一套通吃”
- **人工调参**成本高、周期长、易出错





PID控制器的局限性：为何我们还需要更智能的机制

- **只能被动响应，缺乏预测性**
- 无法预判节假日高峰、羊毛党爆量等突发事件
- 缺少“提前踩刹车”的参数提前自适应机制
- “大事件”前要**手动切换**一套 PID 参数

PID控制器的局限性：为何我们还需要更智能的机制

- **应对复杂性不足，缺乏策略演化能力**
- **多目标**平衡难：体验 vs 成本 vs 留存
- 多用户类型难区分：普通用户 vs 羊毛党
- 非线性波动、历史行为模式等难以建模

PID控制器的局限性：为何我们还需要更智能的机制

极客时间 训练营

- **结论：PID 是好用，但不够聪明**
- 面对复杂的多模型服务系统，我们需要一种能“**学习经验、自动适应、长期优化**”的新机制。

- **引出核心问题：**
- 有没有一种方法，能让系统自己**学会**：
- 什么时候该切换参数?
- 用什么策略既省钱又不伤用户体验?
- 针对不同用户，采取不同对策?

目录

- 模块一：行为主义与控制论
- **模块二：强化学习入门（MDP 与 Q-Learning）**
- 模块三：深度强化学习（DQN）
- 模块四：从强化学习到语言智能体预测控制

强化学习：让控费系统“自己学会最优策略”

极客时间训练营

- **我们真正面临的挑战**

- 传统控制系统关注：
- **控制变量稳定**（如成本、响应速度）
- **不关注长期目标**（如用户满意度、系统效益）

- 但我们的目标是：
- **在不超预算的前提下，最大化长期用户体验与系统收益**

强化学习：让控费系统“自己学会最优策略”

极客时间训练营

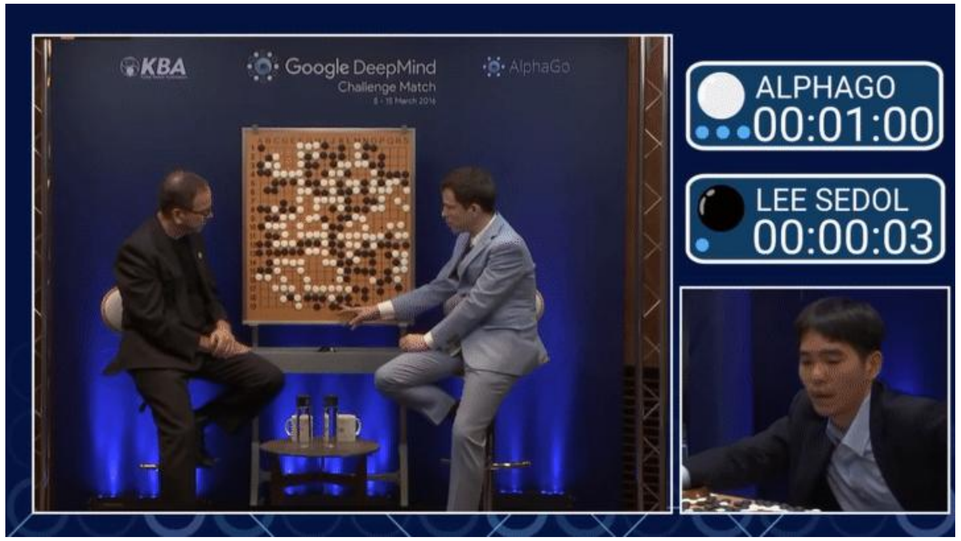
强化学习：让系统“学会”最优行为

强化学习适用于以下复杂场景：

- **缺乏明确标签**（无法直接监督）
- 环境反馈具有**延迟性或不确定性**
- 决策需要考虑**长远**影响（非单步最优）
- 策略难以手工设计，需**端到端优化**

为什么使用强化学习（RL）而非监督学习?

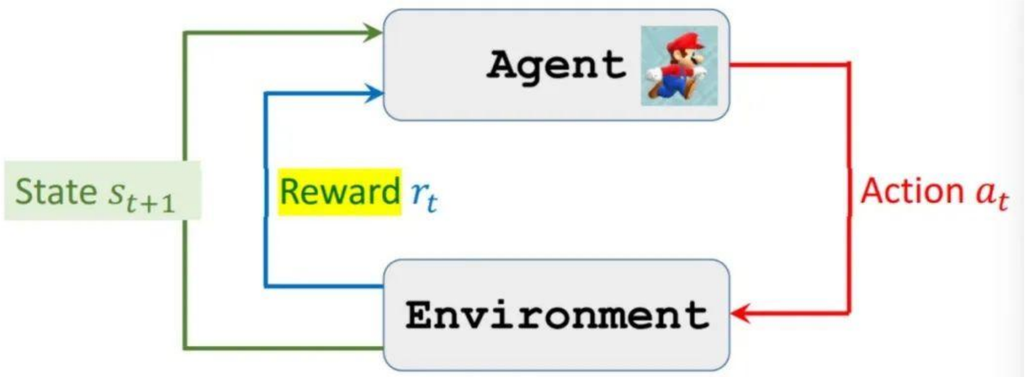
- **强化学习专注于行动选择**
- 强化学习不回答“结果是多少”，而是：**在哪个状态下，采取什么行为能获得长期最优收益?**
- 强化学习直接优化整个“策略”，而非单步输出它关注的不只是“这一刻的反馈”，而是：
- **在整个时间序列中最大化总收益，**学会**权衡当前与未来**的收益
- 可学习**允许短期成本上升，换取长期目标达成**的行为模式



Actor–Environment 架构：让系统在交互中自我学习

极客时间 训练营

- **传统方法的局限**
- 无法定义每一步“正确动作”
- 缺乏明确标签（人也不知道该怎么做）
- 环境反馈延迟、不确定、难归因



- **强化学习的关键突破：无需标签，自主学习**
- 强化学习采用 **Actor–Environment** 架构：
- **智能体**（Agent）与**环境**不断交互，根据**奖励**（Reward）更新策略
- 不再需要“人来告诉你该做什么”
- 只需能衡量行为好坏的**奖励信号**（reward）

Actor–Environment 架构：让系统在交互中自我学习

- **学习信号来自哪里?**
- 强化学习依赖的不是标签，而是**环境反馈**：

行为结果	奖励（Reward）
用户体验提升	+1
成本超预算	-1
用户中途流失	-5
达成留存目标	+10

- 这些反馈信号即可驱动策略不断优化：　　**Reward 就是强化学习的“标签”**
- **而且奖励可以直接和业务 KPI“挂钩”**

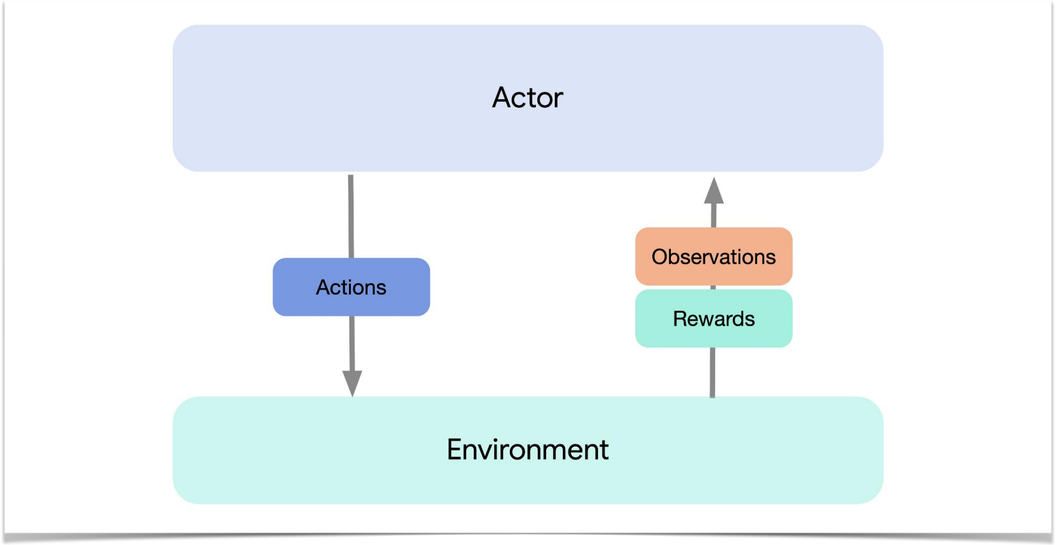
Actor–Environment 架构：让系统在交互中自我学习

- **举例说明：为什么RL能做传统方法做不了的事**
- **传统方法：**“给我一个样本，我告诉你正确答案”
- **强化学习**
- “你只需告诉我做得好不好，我自己学会怎么做得更好”
- 例如：
- 没法标注“**是否该限额**”的每个节点**决策**
- 但可以**定义**：“如果导致预算爆掉，则行为很糟糕（负**奖励**）”

Actor–Environment 架构：让系统在交互中自我学习

- **交互式学习，打破标签依赖：**强化学习不依赖于人类经验，而是靠与环境的**闭环交互**
- **无需标签 → 降低人工成本**
- **环境反馈驱动 → 适配复杂系统**
- **动态学习 → 策略随环境演化而进化**

- **Actor–Env 架构让系统具备“边做边学”的能力，逐步逼近最优策略**



强化学习五要素框架：推理控费场景的精准映射

五要素结构：通用RL框架中的核心角色

元素	强化学习术语	推理控费中的具体含义
Agent	智能体	控制模型切换和分流的策略模块
State	当前状态	系统运行指标：如 OSI 值、用户负载、预算进度等
Action	动作选择	决策：是否切换模型？使用大模型 or 小模型？
Reward	环境奖励	综合指标：用户体验得分 - 成本惩罚
Policy	策略函数	状态到动作的映射规则，即“看到什么，该如何行动”

Q-Learning：从试错中学会控费策略

- **核心思想：用试错交互学习长期价值**
- 从 PID 到 Q-Learning 本质上是从 **误差优化** 思维到 **期望优化** 思维的转化。
- 通过与环境反复交互，学习在每种状态下采取不同动作所能带来的**长期累积收益**。

- 这种收益被称为 **Q值（Quality Value）**，定义为：

“在状态 s 下采取动作 a，未来总共能获得多少奖励”

- Q-Learning 不依赖人工标签或预设规则，而是完全基于环境反馈，自主修正行为偏好、优化策略。

Q-Learning 三要素：在模型控费中的精确映射

极客时间 训练营

- **状态（State）：系统运行的上下文快照**
- 状态用于描述当前系统所处的环境。对于推理成本控制任务，状态可定义为：

$$s=[OSI,用户等级,时间进度]$$

状态变量	含义
OSI（On-Schedule Indicator）	当前成本使用进度 / 预算应使用进度，反映是否超支或滞后
用户等级	区分普通用户与高价值用户（如VIP、付费用户）
时间进度	当前处于月初、中期或临近月末的阶段

- 状态变量可进行离散化处理，构建有限状态空间，提升训练效率

Q-Learning三要素：在模型控费中的精确映射

极客时间 训练营

- **动作（Action）：可选的推理服务策略**
- 在每一个状态下，智能体可以选择如下模型调度动作：

动作	说明	成本	质量
A1	使用 QWen-32B（大模型）	高	最佳
A2	使用 QWen-8B（中模型）	中	中
A3	使用 QWen-8B-int8（小模型）	低	略降

- 这些动作构成智能体的可选决策集合：
  $A=\{A1,A2,A3\}$
- 系统的目标：根据当前状态，选择**收益权衡最优**的动作。

Q-Learning三要素：在模型控费中的精确映射

极客时间 训练营

- **奖励（Reward）：策略优化的方向指引**
- 奖励函数用于评估某一决策的好坏，定义为：

$$Reward=w1\cdot体验得分+w2\cdot留存贡献-w3\cdot成本控制$$

- **体验得分**：响应内容是否完整、准确（如结构性、token使用充分）
- **留存贡献**：是否触发续费、高价值行为等后续转化指标
- **成本控制**：对 OSI 接近 1.0 的目标做了正面还是负面贡献。
- 策略目标：在预算进度（OSI≈1.0）受控的前提下，最大化关键用户体验与留存收益

Q-Learning：从试错中学会控费策略

策略目标：学习一个最优行为映射函数

- 在 Q-Learning 中，我们希望学习一个策略函数

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

- 即：
- 在每一个状态 s 下，选择能带来**长期最大回报**的动作 a。
- 这个策略函数不是硬编码，而是从交互中“试”出来的

Q表（Q-table）：经验的行为记忆库

极客时间训练营

- Q表是一个二维表格，记录所有**状态–动作对的长期 KPI 收益估计**：

$$Q: S \times A \rightarrow R$$

状态（OSI, 用户等级, 时间段）	A1: Qwen-32B	A2: Qwen-8B	A3: Qwen-8B-int8
(高, 高, 月初)	8.2	7.4	3.5
(中, 普通, 月中)	4.1	6.0	6.5
(低, 高, 月末)	2.3	3.1	6.8

- 每一格 Q(s,a) 表示：**在状态 s 下选择动作 a 的长期价值**
- Q表通过持续更新，逐步形成对不同情境下的“最优动作偏好”

Q-Learning：从试错中学会控费策略

每次与环境交互后，Q表会更新某个状态-动作对的估值：

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[\left\{r + \gamma \max_{a'} Q(s',a')\right\} - Q(s,a)\right]$$

项	工程含义
Q(s,a)	**当前估计**：系统原本“以为”这个动作在此状态下值这么多
r	**当前的即时回报**：本次实际的控费结果（如体验好坏、成本高低）
max Q(s', a')	**对未来所有回报的折现估算**：如果现在进入新状态，**下一步最聪明的做法能带来多少回报**
r+γ· maxQ(s′,a′)	**总回报估算**：当前行为“应得的”长期回报（包含现在 + 将来）
α	学习率：控制新经验对旧判断的修正幅度，值越大越“相信最近发生的事情”
γ	折扣因子：越接近1越重视**长期回报**，越接近0越偏向**短期反馈**

为什么 Q-Learning 要用

总回报估算−当前估计=[r+γ·maxa′Q(s′,a′)]−Q(s,a)

作为更新信号（也叫**TD误差**，Temporal Difference Error）？

你原来以为它值 Q(s,a)，但现在看来它值更多→ 就要补上这个差值 δ，让“想象”贴近“现实”

Q-Learning：从试错中学会控费策略

- **算法直觉：一步步逼近“真实行为价值”**

- 更新过程的本质是：**当前估计** ← **当前奖励** + **未来最优动作的估值**

- 每一次交互（状态→动作→奖励→新状态）都会让 Q 值更接近真实的长期回报，从而逐步逼近最优策略。

Q-Learning vs PID：智能调度的能力对比

- **举个例子：模型选择对比策略**

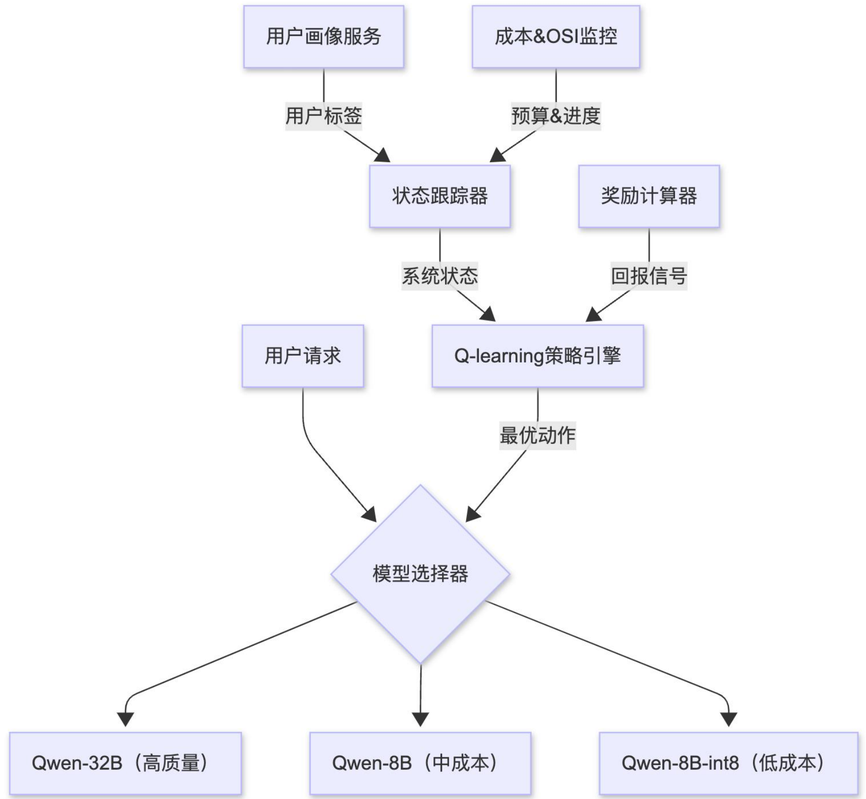
状态（示例）	PID反应	Q-Learning行为
月初 + 用户用量低	降低大模型占比	提高体验以促转化
月中 + 成本过高	直接限流	限流 + 判断用户价值再决策
月末 + OSI不足	被动调增比例	主动集中资源确保关键用户体验
VIP用户	无区分	主动分配优质模型 + 延长对话窗口

- **总结：Q-Learning 是一种“经验驱动的智能控制范式”**
- 不再纠结“当前误差”，而是建构“全局策略”
- 不再只调一个比例，而是决策“何时升级 / 降级 / 分流”
- 从“纠偏器”进化为“策略家”

Q-Learning vs PID：智能调度的能力对比

1. **用户发起请求**
2. **状态跟踪器**获取当前上下文（如OSI、用户等级）
3. **Q-Learning引擎**根据状态选择最优模型（A1/A2/A3）
4. **模型执行响应**
5. **收集反馈信息**（成本、质量、用户行为等）
6. **奖励计算器**生成当前动作的奖励
7. **策略更新**：通过Q值更新改进策略

模块	说明
用户请求入口	触发一次模型调用需求
模型选择器	按照当前策略（由RL给出）选择最合适的模型
Q-Learning引擎	输入当前状态，输出最优动作（模型选择）
状态跟踪器	实时记录如OSI进度、时间阶段、用户标签等
奖励计算器	根据响应后的成本、质量、用户行为等计算奖励
用户画像服务	提供用户身份、历史、留存等级等信息
成本与进度监控	提供预算消耗和OSI执行进度数据



实际部署效果：智能策略带来的全局优化

- **预期30天 A/B 测试结果**

关键指标	PID控制	Q-Learning策略	相对提升
平均用户体验得分	0.82	0.85	**↑ 3.7%**
进度延迟（OSI<0.95）比例	18.3%	9.1%	**↓ 50.3%**
VIP用户满意度	76%	89%	**↑ 13.0%**

- 强化学习策略在“成本合规性、进度保障、体验质量”三维度均显著优于传统PID
- **用户续订率提升背后的策略差异**

时段	PID行为	Q-Learning策略
月初	一刀切限流，用户流失高	VIP优先，低价值场景保守降级
月中	规则过度反应，服务波动	稳定策略控制，逐步回正成本
月末	补救式提档，但为时已晚	预判式预算腾挪，集中保证关键服务

从Q表到Q网络：走向深度强化学习的下一步

- **Q-Learning的能力边界**
- 虽然 Q-Learning 在离散状态空间中表现出色，但在真实业务中，我们面临更高维、更动态的环境状态：

问题	说明
状态空间爆炸	状态维度（如：用户画像、时间进度、预算进度、行为上下文）组合数呈指数增长
Q表不可扩展	传统 Q-learning 使用 Q-table 显式存储每个状态-动作对，难以扩展到百万级状态空间
无法泛化	Q值仅对“见过”的状态-动作对有效，不能处理“相似但没见过”的场景
无法处理连续状态	如时间、预算、token密度等连续变量，无法离散化建表处理

- 由 DeepMind 提出，**DQN = Q-Learning + 深度神经网络**

特征	DQN带来的优势
多模态状态（用户、时间、预算）	支持向量/embedding输入、端到端学习
高维动作空间（如多模型、多层级）	可引入动作嵌入与策略结构
策略泛化（新用户、新预算周期）	网络结构具备泛化能力
在线训练与迁移	可结合经验回放与冷启动策略训练

目录

- 模块一：行为主义与控制论
- 模块二：强化学习入门（MDP 与 Q-Learning）
- **模块三：深度强化学习（DQN）**
- 模块四：从强化学习到语言智能体预测控制

Q-Learning的能力边界：当Q表不再够用

极客时间 训练营

Q表的致命问题：状态组合爆炸

Q表虽然直观易用，但它只能应对**状态空间离散、维度较小**的任务。

在真实控费系统中，状态往往包含：

维度	示例
OSI	连续值（如0.41、0.78）
用户等级	多级分层、动态变化
时间进度	连续时间、非线性影响
用户画像	行为、活跃度、是否续订过等特征嵌入

→ 这导致状态组合总数呈**指数增长**：

状态数=离散段数 dot 维度数⇒Q表大小迅速失控

Q-Learning的能力边界：当Q表不再够用

Q表的问题总结

问题	描述
状态爆炸	状态变量稍多就需要上万甚至上亿条Q值记录
不支持连续状态	连续值必须手动离散，易丢信息
不能泛化学习	学到的是“某一具体状态”的动作偏好，而不是“相似状态”的泛化策略
更新不通用	每次学习只影响当前状态-动作对，无法推广到相邻场景
冷启动难	未见过的状态-动作对 Q 值为0，策略无法做出合理决策

控费场景中的现实挑战

一个真实推理任务中的状态可能是：

OSI = 0.74、时间 = 15:03、第3轮对话、用户最近刚转为付费、token平均响应长度增加

Q表根本无法覆盖这类复杂上下文，更无法泛化策略到“类似但未见过”的状态。

Q-Learning的能力边界：当Q表不再够用

极客时间训练营

直觉类比：Q表像“离散字典”，而不是“认知函数”

类比	Q表	人类
记忆方式	记录每一项“该怎么做”	学习模式，遇到类似情境能推理
表达能力	明确但僵硬	泛化但可适应
缺陷	状态多就崩	情境复杂也能应对

为了解决这些问题，研究者提出了一个突破性方案：

用一个**神经网络**来近似 Q(s,a)，不再用显式Q表记录每一对状态–动作值

这就是 DeepMind 提出的**DQN（Deep Q-Network）**。

DQN = Q-learning + 神经网络

能够处理高维状态、连续变量、未见过场景，并具备泛化能力

DQN：让强化学习适配真实业务复杂度

极客时间训练营

DQN 架构：从“查表”到“函数拟合”

DQN 如何工作?

DQN 不再依赖离散的表格，而是用一个深度神经网络来直接拟合状态到动作价值的函数。

输入状态 (Input State):

系统将当前的高维、连续状态 s（如 OSI、用户等级、对话轮次等）作为一个向量，直接输入给神经网络。

网络拟合 (Network Approximation):

神经网络 Q(s, a; θ) 充当一个强大的函数逼近器，通过其可训练参数 θ，学习状态特征与长期价值（Q值）之间的复杂非线性关系。

输出决策 (Output Decision):

网络一次性计算并输出当前状态 s 下，所有可选动作（如 A1: 高级模型, A2: 标准模型, A3: 降级模型）的预估 Q 值。

智能体（Agent）遵循贪心策略，选择 Q 值最高的动作执行。

DQN：让强化学习适配真实业务复杂度

DQN 的核心优势：

处理高维连续状态 (Handles High-Dimensional & Continuous States):

- 无需对状态进行离散化。像 OSI=0.735 或 对话平均 token=128.5 这样的精确浮点数可直接作为输入，保留了完整信息。

强大的泛化能力 (Powerful Generalization):

- 对于训练时从未见过的状态组合，网络也能根据已学习到的数据模式，给出合理的 Q 值估计。
- **Q 表:** 没见过 = 不知道怎么办
- **DQN:** 没见过 ≈ 和某个见过的相似，我猜应该这么办

DQN 的训练秘诀：经验回放与目标网络

直接用 Q-Learning 的方式训练神经网络非常不稳定。DQN 引入了两项关键技术来解决这个问题。

1. 经验回放 (Experience Replay)

问题： 智能体在环境中连续交互产生的数据（(s, a, r, s')）前后高度相关，直接用于训练会违反神经网络“独立同分布 (IID)”的假设，导致训练效率低、模型收敛难。

解决方案： 建立一个“经验池”（Replay Buffer），将智能体的所有经历 (状态, 动作, 奖励, 下一状态) 储存起来。训练时，不直接使用刚产生的经验，而是从经验池中随机抽取一小批（mini-batch）数据进行训练。

优势：

打破数据相关性： 随机采样打乱了数据的时间顺序，使训练样本更接近独立同分布，训练过程更稳定。

提升数据利用率： 每一次的宝贵经历都可以被反复利用，提升了学习效率。

DQN 的训练秘诀：经验回放与目标网络

2. 固定目标网络 (Fixed Target Network)

问题： Q-Learning 的更新目标 r+gammamax_a′Q(s′,a′) 是由正在更新的网络自己计算的。这相当于“用一个移动的目标来指导更新”，容易导致训练过程震荡或发散。

解决方案： 使用两个结构完全相同但参数不同的网络：

策略网络 (Policy Network, 参数 theta): 负责在每一步根据当前状态选择动作，并且在训练中实时更新其参数。

目标网络 (Target Network, 参数 theta⁻): 专门用于计算更新目标中的 Q(s′,a′) 部分。 它的参数是定期从策略网络复制而来，在两次复制之间保持固定。

优势：

稳定学习目标： 将更新目标 y=r+γ max_a′Q(s′,a′;θ⁻) 在一段时间内固定下来，避免了“追逐自己尾巴”的问题，让学习过程更加平稳。

DQN 的训练秘诀：经验回放与目标网络

DQN 的损失函数

综合以上两点，DQN 的目标是让“策略网络”的预测值 Q(s,a;theta)，不断逼近由“目标网络”计算出的、更稳定的目标值 y。

损失函数 (Loss Function):

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\underbrace{\left(r + \gamma \max_{a'} Q(s', a'; \theta^-)\right)}_{\text{目标值 (Target) by Target Network}} - \underbrace{Q(s, a; \theta)}_{\text{预测值 (Predicton) by Policy Network}}\right]^2$$

我们将 DQN 的理论框架应用于大模型推理控费，形成一个端到端的智能决策与学习闭环。

系统运行步骤

1. **感知状态 (Perceive State):** 系统捕捉当前的用户与平台信息（如 OSI、用户等级、对话历史），构建成一个高维状态向量 **s**。
2. **决策动作 (Decide Action): 策略网络** Q(s,a;theta) 接收状态向量 **s**，快速计算出所有可选动作（A1, A2, A3...）的 Q 值，并选择 Q 值最高的动作 **a** 执行（例如：决定本次调用使用“标准版模型”）。
3. **环境交互 (Interact with Environment):** 系统执行动作 **a**，为用户提供服务。服务完成后，环境会产生新的状态 **s'**，并根据成本、用户反馈等计算出本次动作的**奖励 r**。
4. **存储经验 (Store Experience):** 将这次交互的完整记录 (s, a, r, s') 作为一个“经验”，存入**经验回放池**。
5. **学习与更新 (Learn & Update):**
    1. 训练模块从经验回放池中**随机抽取**一批经验数据。
    2. **目标网络** Q(s′,a′;θ−) 计算出这批数据的稳定目标 Q 值。
    3. 系统计算“目标 Q 值”与“策略网络预测 Q 值”之间的损失（Loss）。
    4. 通过反向传播算法，更新**策略网络**的参数 θ，使其预测越来越准。
    5. 每隔一定步数，将策略网络的参数 θ 复制给目标网络 θ−，完成一次迭代。
6. 这个循环不断进行，DQN 系统通过持续的自我博弈与学习，自动进化出在“成本”与“体验”间取得最佳平衡的动态调度策略。

DQN策略演化过程：从随机试错到智能控费

初期阶段：策略随机，效果不稳

智能体尚未积累经验

模型选择基本靠猜：大模型用得多，体验好但容易**爆表**

奖励函数反馈明显：高成本 = 惩罚，低体验 = 惩罚

典型现象：

- 重度用户多次爆表
- 轻度用户体验被压低

DQN策略演化过程：从随机试错到智能控费

中期阶段：策略逐步“有规律”

系统开始识别出**不同用户、不同状态的最优策略**

对高风险状态（如 OSI > 0.8）倾向选择小模型

对低风险状态（如 OSI < 0.4）大胆启用大模型

行为特征：

- 智能体开始分人分场景做出差异化响应
- 分流权重逐渐稳定，不再剧烈震荡

DQN策略演化过程：从随机试错到智能控费

收敛阶段：形成稳定、高效策略

多数用户在预算范围内获得尽可能好的体验

系统学会了“提前控本 + 动态让利”的平衡机制

样例策略：

- 普通用户全天保持中模型，体验成本均衡
- 重度用户高峰前提前降级，小模型保障限额不爆
- 低频用户允许高体验突发透支，提升满意度

从 Q-Learning 到 DQN 的进化

- **解决了什么问题?**
  - DQN 用神经网络替代 Q 表，解决了状态空间组合爆炸的问题，使其能够处理真实世界中高维、连续的复杂状态。
- **核心优势是什么?**
  - **泛化能力**：能够对未曾见过的状态做出合理决策。
  - **训练稳定**：通过“经验回放”和“固定目标网络”两大创新，解决了传统强化学习在非线性函数逼近下的训练不稳定性。
- **下一步是什么?**
  - DQN 已经非常强大，但它主要处理离散动作空间，并且在某些场景下仍然存在局限。
  - 当我们的决策变得更复杂，甚至需要生成自然语言本身作为一种“动作”时，我们需要更前沿的智能体技术。

目录

- 模块一：行为主义与控制论
- 模块二：强化学习入门（MDP 与 Q-Learning）
- 模块三：深度强化学习（DQN）
- **模块四：从强化学习到语言智能体预测控制**

从微观调控到宏观决策：RL + 语言智能体的协同演进

问题回顾：

强化学习擅长做**毫秒级别**的“用不用大模型”决策
但整个大模型服务系统，还需要做更“全局、长周期”的管理

系统未来能力目标：

层级	控制类型	决策对象
微观（秒级）	Token 层实时控制	当前请求用哪个模型？分流比例？
宏观（天级）	策略层全局运营	什么时候该限额？如何做节假日预案？

从微观调控到宏观决策：RL + 语言智能体的协同演进

微观调控：强化学习（RL）的职责

任务：**每次请求做出最优切换决策**

优势：

- 响应快、低延迟
- 自适应用户行为变化
- 适合“短期反馈目标优化”（如当前体验 × 当前成本）

宏观决策：语言智能体（ReAct Agent）的职责

任务：**跨时间、跨场景做全局分析与策略生成**

能力：

- 理解业务目标、周期变化、异常趋势
- 制定多天多人群的“控费与体验策略”
- 生成 A/B 流控方案、节假日扩容预案等

从微观调控到宏观决策：RL + 语言智能体的协同演进

协同机制：微观与宏观的智能对接

模块	职责
ReAct 智能体	定期生成宏观策略、风险分析报告
RL 智能体	实时执行策略、根据反馈微调权重
中控平台	统一协调两者，支撑系统闭环运行

工业智能系统形成“全链路闭环”：

语言智能体 **负责“看得远”**，强化学习 **负责“调得准”**

ReAct 智能体：大模型控费系统的“宏观大脑”

极客时间训练营

什么是 ReAct Agent?

ReAct = Reasoning + Acting
是一种结合**大语言模型能力**与**工具调用能力**的智能体架构
能够像运营人员一样进行**分析、规划、执行**

ReAct 智能体的核心结构

组件	功能描述
自然语言理解引擎	理解策略需求、系统日志、用户行为和业务目标
外部工具接口	调用数据 API、监控平台、流控平台、告警系统等
多步推理模块	支持 Chain-of-Thought 推理与策略生成
报告生成模块	生成运营分析报告、策略建议、扩容计划等

ReAct 智能体：大模型控费系统的“宏观大脑”

宏观控费任务中的应用能力

高峰预判与预案生成

检测节假日临近（如“双11”）

回溯历史用量高峰 → 自动生成多套流控策略

对接 RL 控制器 → 调整限额和模型使用阈值

ReAct 智能体：大模型控费系统的“宏观大脑”

极客时间 训练营

宏观控费任务中的应用能力

异常行为智能溯源

某用户短时间用量暴涨?

自动比对历史行为模式，识别“羊毛党”

动态调高该用户的 OSI 降级阈值

ReAct 智能体：大模型控费系统的“宏观大脑”

宏观控费任务中的应用能力

多目标策略仿真

同时考虑成本、体验、留存、用户等级

运行 A/B 流控模拟，选择长期收益最优方案

ReAct 智能体：大模型控费系统的“宏观大脑”

宏观控费任务中的应用能力

周期性运营分析

自动汇总本周控费执行情况

标注爆表用户、高价值群体、策略失效区域

输出结构化报告供人工审阅或系统自用

ReAct 智能体：大模型控费系统的“宏观大脑”

与 RL 智能体的协同方式

ReAct 智能体职责	RL 智能体职责
策略制定、预案生成、风险分析	具体执行、实时反馈、参数微调
探测宏观趋势、控制策略升级	快速响应微观状态变化、精细化调度

整体效果：实现战略–战术级联闭环

ReAct 负责“战略方向”，RL 负责“战术执行”

AI驱动的全链路控费运营：智能体协同实战案例

极客时间 训练营

业务场景设定：双11大促临近，系统需防爆表、保体验

阶段一：ReAct 智能体启动“宏观预判”

节日感知：

- 检查业务日历，发现大促即将到来
- 历史数据检索，确认“双11”期间流量预计激增 300%+

数据分析与策略规划：

- 汇总往年用户行为、爆表原因、模型使用分布
- 针对不同用户群（普通 / 羊毛党 / VIP）生成多套控费策略：

策略下发：

- 将策略映射为具体参数，自动写入 RL 控制器配置表

用户类型	策略生成示例
羊毛党	提前限额、OSI阈值提前触发、禁止大模型
普通用户	保持当前策略，轻度动态调整
高价值用户	宽松额度、容忍短期透支、体验优先

AI驱动的全链路控费运营：智能体协同实战案例

阶段二：RL 智能体实时动态执行

按用户类型与当前状态，**毫秒级决策模型选择与切换**

若发现某用户调用量异常上升：

- 实时提升小模型分流比例
- 动态调整用户 OSI 预警阈值

若系统总体负载临界：

- 全局降低大模型使用概率，实现削峰填谷

AI驱动的全链路控费运营：智能体协同实战案例

阶段三：ReAct 持续监控与策略迭代

每 15 分钟汇总一次关键指标：

- 爆表率、用户满意度、token成本走势

自动判断策略效果是否达标：

- 若不达标 → 修改策略参数 → 重新下发 RL 控制器

闭环效果

目标	成果
控本目标	总爆表率控制在 1.8%，远低于预警线
体验目标	92% 用户无感知模型切换，满意度维持在 90+ 分
运维效率	无需人工盯盘，全链路自动监控与动态调控完成

总结：

RL 智能体：**“每秒调度”的控费决策器**
ReAct 智能体：**“全天候运营”的智能参谋长**

两者协同，实现了 AIGC 系统的真正**自主管理与自我优化**

从控制器到RL，再到语言智能体

- 行为主义和控制论不仅奠定了现代AI智能体的理论基石，也让我们能用“反馈–学习–自适应”的统一视角设计和进化复杂产业AI系统。这正是AI系统“从简单反应，到复杂适应，再到自主运营”的核心动力。

阶段	技术代表	控制范式	核心能力与价值
闭环反馈控制	PID控制器	被动调节	自动应对偏差、提升运维效率，适用于稳态、规则清晰场景
自适应智能体	RL/DQN	端到端学习	动态优化体验-成本平衡，能适应多用户、多场景、复杂目标
语言智能体预测控制	ReAct Agent	主动感知与规划	复杂信息融合、主动风险预警、宏观运营决策，替代/辅助人类运营

- 从行为主义 → 到控制论 → 到强化学习 → 到智能代理系统
  AI 系统走向具备**持续学习、策略演化、全局感知**的真正“自我调节者”

- “让AI替代重复、复杂、动态的决策，让人类专注于创新与价值创造——这正是AI智能体与产业共进的核心使命。”

推荐阅读

- 《控制论与科学方法论》金观涛，华国凡.
- The Second Half. Yao, Shunyu. https://ysymyth.github.io/The-Second-Half/

THANKS