浙江大学

本科实验报告

课程名称：数据要素市场

学　　院：竺可桢学院

系：所在系

专　　业：计算机科学与技术

姓名学号：张晋恺 3230102400

姓名学号：汪昕 3230101888

姓名学号：施兴睿 3230102392

指导教师：刘金飞

2025 年 7 月 22 日

目录

1 INTRODUCTION

1.1 项目概述

这篇论文的核心是解决一个在线数据定价问题：卖家拥有大量同质数据点（都从一个独立同分布的分布中采样得到），希望通过设定价格曲线来出售数据以最大化长期收益；买家分为 m 种类型，每种类型都有一个估值曲线 $v_i(n)$，表示购买 n 个数据点获得的价值。

它讨论的问题与一般的的多臂老虎机以及完全信息反馈问题不同之处在于这是一个非对称反馈问题：

- 在随机设置中，卖家事先不知道买方的类型分布。
- 在对抗性设置中，卖家事先不知道买方类型的序列。

买家知道自己的类型分布以及卖家设置的价格曲线。当买家购买后，会向卖家透露其类型信息，若买家决定不购买，则不会向卖家透露自己的任何类型信息。

卖家在设计价格曲线时需要进行一定的权衡：

- **利用**：如果价格曲线较高，可能可以增加即时收益，但也可能会导致买家不购买，在无法获得任何收益的同时也无法获取买家的类型信息，这对于后续最大化长期利益不利。
- **探索**：如果价格曲线较低，可能会吸引更多买家购买，虽然能获取更多的类型信息，但可能会导致卖家的收益降低。

为了使这个问题可解，论文引入了以下假设（在实际数据和机器学习应用中通常成立）：

- **单调性**，M：买方估值曲线 $v_i(n)$ 是非递减的（即数据越多价值越大）
- **有限类型**，F：买方类型 m 的数量有限
- **平滑性**，S：对于所有的 $n, n' \in [N]$，估值的变化满足 $v_i(n+n') - v_i(n) \leqslant \frac{L}{N} n'$。这表明买方的估值在购买少量额外数据点时不会剧烈改变。

- **收益递减**，D：存在某个 $J > 0$，使得对于所有类型 $i \in [m]$ 和所有 $n \in [N]$，有 $v_i(n+1) - v_i(n) \leqslant \frac{J}{n}$。
  这意味着额外数据点带来的边际价值会随着已拥有数据量的增加而递减，这与许多机器学习准确性曲线的特性一致。

论文的贡献在于提出了新的离散化方案和在线学习算法：

1. 论文证明了在只有 m 种类型的情况下，最优定价曲线是“简单”的，即任何非递减的价格曲线都可以用一个“m 步”的阶梯跳跃价格曲线 $\bar{p}$ 来近似且其产生的预期收益至少相同，这极大地缩小了需要考虑的价格曲线空间。

2. 论文还对不同的在线设置（随机设置与对抗性设置）设计了相应的算法（基于 UCB 和基于 FTPL），分别实现了 $\tilde{O}(m\sqrt{T})$ 和 $\tilde{O}(m^{3/2}\sqrt{T})$ 的（期望）遗憾界。

虽然总体计算复杂度可能随买方类型数量 m 增加而呈指数级增长，但论文提供的算法能保证在类型数量有限但数据集规模庞大时的计算可行性。

2 THEORY

2.1 理论部分

2.1.1 基于 UCB 的动态定价算法

在随机设置下，买方类型服从一个固定但未知的分布 q，其中面临的挑战包括：

1. 价格曲线空间巨大：价格曲线是一个函数，直接搜索所有可能的函数显然不可行，即使进行简单的离散化，价格曲线的数量也可能巨大，导致 UCB 算法的遗憾值（regret）对价格曲线空间的大小 $|\bar{\mathcal{P}}|$ 呈指数依赖，性能很差。

2. 非对称反馈：卖方只在买方购买数据时才能知道买方的类型，如果买家没有购买，卖方无法得知当前买方的类型信息。

设计思想

论文对这两个挑战分别提出了解决的思路和方案：

1. 离散化方案：

   - 论文首先证明了对于 m 种买家类型，任何非递减的价格曲线都可以用一个“m 步”的价格曲线 $\bar{p}$ 来近似，大大缩小了需要考虑的价格曲线空间。
   - 论文还对估值空间（价格的范围）和数据数量空间进行了精细的分层离散化，尤其是在价格较低或数据量较少时更加密集，以捕捉重要的变化。
   - 最后论文还进一步结合曲线的单调性、平滑性、收益递减性等，进一步缩小了离散化曲线集合 $\bar{\mathcal{P}}$ 的大小。

   经过上述的离散化方案，虽然 $|\bar{\mathcal{P}}|$ 仍然可能很大，但它使得后续的 UCB 算法可以在这个有限集合上运行。

2. 构建 UCB：面对非对称反馈问题，论文的创新之处在于没有为每个价格曲线维护 UCB，而是为买家类型分布 q 维护一个 UCB，然后再把 q 的 UCB 转化为收益的 UCB。

   - 在第 1 轮中，算法选择一个 0 价格曲线（任意数量数据的价格都为 0），以确保所有买家都会购买数据并透露其类型，从而获得初始的类型反馈信息。
   - 记录在过去 t 轮中类型 i 的买方在理论上能够从已选择的价格 p_τ 中获得非负效用（即能进行购买）的次数 $T_{i,t}$
   - 在类型 i 会购买的那些轮次中，统计类型 i 实际出现的经验频率 $\bar{q}_{i,t}$，然后利用 Hoeffding 不等式构建对真实频率 q_i 的乐观估计上界 $\hat{q}_{i,t}$
   - 最后利用卖方已知的估值曲线 $v_i(n)$ 和刚刚计算出的类型频率 UCB $\hat{q}_{i,t}$，计算出任意价格曲线 p 的预期收入的 UCB $\widehat{\mathrm{rev}}(p) = \sum_{i=1}^{m} \hat{q}_{i,t} \cdot p(n_{i,p})$。算法在每轮中会选择离散化价格集合 $\bar{\mathcal{P}}$ 中预期收入 UCB 最大的价格曲线 p_t 作为当前轮次的定价曲线。

证明思路

这个算法的遗憾可以被分解为离散误差和估计误差两部分

1. 首先证明在上述的离散化方案中离散误差可以被近似到 $O(\varepsilon)$ 的范围内。

2. 接着使用 Hoeffding 不等式证明 $|q_i - \hat{q}_{i,t}| \leqslant O(\sqrt{\frac{\log T}{T_{i,t}}})$，即 $\hat{q}_{i,t}$ 对 q_i 估计误差以 $\sqrt{\frac{\log T}{T_{i,t}}}$ 的速度衰减。

3. 最后可以证明估计误差不超过 $\tilde{O}(m\sqrt{T})$，计算两部分之和得到的遗憾界也不超过 $\tilde{O}(m\sqrt{T})$

2.1.2 基于 FTPL 的动态定价算法

在对抗性设置中，买家类型的序列是由对手选择的，这相当于总是遇见最坏的情况，这要求设计一个对任意序列输入都有良好表现的算法。

设计思想

FTPL 通常用于完全信息问题，论文修改了 FTPL 中的“奖励” $r_t(p)$ 的定义来适应这个非对称反馈问题中的情况。

- 发生购买时：卖家会得知买家的类型 i_t，此时所有价格曲线 p 的奖励为在这一类型和当前价格 p 下的真实收益 $r_t(p) = p(n_{i_t,p})$。

- 不发生购买时：卖家不知道本次买家的具体类型，但卖家知道哪些类型的买家一定会购买，这些类型的集合记为 S_t，因此本次买家的类型一定属于 S_t^c。
  算法此时将奖励定义为在价格 p 下，所有属于 S_t^c 的类型的收益之和（相当于一个上界）：$r_t(p) = \sum_{i \in S_t^c} p(n_{i,p}) \geqslant p(n_{i_t,p})$
  这么做在直观上的理解为给出一个更大的奖励来鼓励对这些不愿意购买的类型的探索，从而获取更多的类型信息。

在算法的每一轮中，我们会选择累计奖励加上一个扰动值最大的价格曲线 $p_t = \arg\max_{p \in \bar{\mathcal{P}}} \sum_{\tau=1}^{t-1} r_\tau(p) + \theta_p$，其中扰动 θ_p 采样于指数分布 $\theta e^{-\theta x}$。

证明思路

总遗憾同样可以分解为两部分：离散误差和算法误差。前者是将连续价格曲线空间离散化为 $\bar{\mathcal{P}}$ 带来的误差，后者是由于对抗性设置下的算法在进行决策时带来的遗憾，我们更多的关注后者。

FTPL 算法通常用于完全信息的问题，在该论文设置的对抗性的条件下，我们面对的不是一个完全信息的环境，而论文所构造的“收益”能够在买家不透露类型信息（即不购买）的情况下，能使用一个上界来替代真实的收益（因为 $\sum_{i \in S_t^c} p(n_{i,p}) \geqslant p(n_{i_t,p}),\ i_t \in S_t^c$），所以也能为算法提供鼓励其探索这一买家类型的信息。

FTPL 算法的遗憾界限通常会包含一个扰动参数 θ，论文给出了 $O(m^2\theta T + \theta^{-1}(1+\log|\bar{\mathcal{P}}|))$ 的遗憾界限。我们可以通过优化 θ 的选择来最小化遗憾值的最终上界：令 $\theta = \sqrt{\frac{1+\log|\bar{\mathcal{P}}|}{m^2 T}}$，则最终的遗憾界限为 $\tilde{O}(m\sqrt{T\log|\bar{\mathcal{P}}|})$。又因为离散后的规模为 $|\bar{\mathcal{P}}| = \tilde{O}\left(\left(\frac{J}{\varepsilon^3}\right)^m\right)$，所以最终的遗憾界限为 $\tilde{O}(m^{3/2}\sqrt{T})$。

3 EXPERIMENTS

3.1 实验部分 - UCB

3.1.1 基本结构

实验主要分为三个主要的 python 文件

- UCB.py: 主程序，负责调用其他模块
- utils.py: 工具函数，负责处理数据和模型
- visualization.py: 可视化函数，负责可视化结果

接下来依次介绍每个文件的实现思路

3.1.2 utils.py

估值函数

首先需要为玩家生成估值函数，其数学表达式为

$$v(n) = C * (1 - e^{-\beta * n})$$

其中，C 是估值函数的最大值，β 是估值函数的衰减系数。
这个函数当 n 趋近于无穷大时，$v(n)$ 趋近于 C，当 $n = 0$ 时，$v(n) = 0$。
而且满足 $v(n)$ 随 n 的增大而增大，并且增速越来越小，边际收益递减。
然后根据玩家的类型，生成不同参数的估值函数，用于模拟不同玩家的行为。
具体来说通过 base_rate 和 max_value 两个参数来控制估值函数的形状。

生成价格曲线集合

根据论文中的算法 2，生成一个离散化的价格曲线集合，而且是 $m-step$ 的，即跳跃次数不超过 m 次。

算法 2 的实现主要分为三个步骤：

步骤 1：构建数据空间网格 N_D 首先构建数据空间网格，用于离散化数据点数量 n：

- 设定阈值 $threshold\_n = \lfloor \frac{2Jm}{\epsilon^2} \rfloor$
- 对于 $n \leq threshold\_n$ 的数据点，不进行离散化，直接添加到网格中
- 对于 $n > threshold\_n$ 的数据点，采用对数离散化：
  - 计算 Y_i 点：$Y_0 = threshold\_n$，$Y_{i+1} = Y_i \cdot (1+\epsilon^2)$
  - 在相邻的 Y_i 点之间插入约 $2Jm$ 个内点
  - 使用线性插值生成内点：$inner\_points = linspace(Y_i, Y_{i+1}, 2Jm)$
- 最终将 0 也加入网格中，确保包含不购买的情况

步骤 2：构建估值空间网格 W 然后构建估值空间网格，用于离散化价格水平：

- 计算 Z_i 点：$Z_0 = \epsilon$，$Z_{i+1} = Z_i \cdot (1+\epsilon)$
- 在相邻的 Z_i 点之间进行细粒度插值：
  - 计算间隔：$gap = Z_i \cdot \frac{\epsilon}{m}$
  - 计算插值步数：$num\_steps = \lfloor \frac{Z_{i+1}-Z_i}{gap} \rfloor$
  - 使用线性插值生成内点：$inner\_points = linspace(Z_i, Z_{i+1}, num\_steps)$
- 最终将 0 也加入网格中，确保包含免费价格

步骤 3：生成 m-阶梯价格曲线 基于上述两个网格，随机采样生成 m-阶梯价格曲线：

- 首先添加一条” 免费” 价格曲线（全零价格），这是算法 3 第一轮必需的
- 对于每条新的价格曲线：
  - 随机决定阶梯数 $k \in [1, m]$

- 从 N_D 中随机选择 k 个跳变点 $jump\_points\_n$
- 从 W 中随机选择 k 个价格水平 $price\_levels$
- 构建阶梯价格曲线：

$$p(n) = \begin{cases} price\_levels[0] & \text{if } 0 \le n \le jump\_points\_n[0] \\ price\_levels[1] & \text{if } jump\_points\_n[0] < n \le jump\_points\_n[1] \\ \vdots & \\ price\_levels[k-1] & \text{if } jump\_points\_n[k-2] < n \le jump\_points\_n[k-1] \\ price\_levels[k-1] & \text{if } n > jump\_points\_n[k-1] \end{cases}$$

这种离散化方法确保了价格曲线集合的大小是多项式级别的，同时保持了足够的表达能力来近似最优解。

生成估值函数集合

在 UCB 算法中，需要预先计算每个 (价格曲线, 买家类型) 组合的购买决策和收入，这模拟了 Agent 拥有的先验知识。

购买决策计算 对于每个价格曲线 p 和买家类型 i，计算最优购买量：

- 计算所有可能购买量 n 的效用：$u_i(n) = v_i(n) - p(n)$
- 找到效用最大化的购买量：$n^* = \arg\max_n u_i(n)$
- 如果最大效用小于 0，则选择不购买（$n^* = 0$）
- 如果存在多个效用相等的最大值，根据论文规则选择最大的 n

收入计算 基于购买决策计算收入：

- 收入矩阵：$revenues[i,j] = p_j(n^*_{i,j})$，其中 p_j 是第 j 条价格曲线，$n^*_{i,j}$ 是类型 i 买家在价格曲线 j 下的最优购买量
- 购买量矩阵：$purchases[i,j] = n^*_{i,j}$，记录每种组合的购买量

这种预计算方式确保了算法在运行时能够快速查询任何价格曲线和买家类型组合的预期收入，为 UCB 算法的决策提供基础数据。

3.1.3 UCB.py

Config

配置类负责管理实验参数和生成真实的买家类型分布：

- **基本参数：**
  - M_TYPES：买家类型数量（默认 10）
  - N_ITEMS：数据点总数（默认 100）
  - $T\_HORIZON$：总回合数（默认 $M\_TYPES \times 10000$）
  - J：收益递减常数（默认 2.0）
  - ϵ：近似精度参数（默认 0.1）
  - $num\_samples$：生成的价格曲线数量（默认 $\max(7, M\_TYPES + 5)$）
- **真实分布生成：**根据买家类型数量自动生成合理的概率分布
  - 对于 1-4 种类型，使用预设的分布
  - 对于更多类型，使用高斯分布生成后归一化

UCBAgent

实现了论文中算法 3 的 UCB 算法，主要包含以下函数：

初始化

- 初始化内部状态：$t = 1$（当前回合数）
- T_i：类型 i 被”探索”的次数
- N_i：类型 i 被实际观察到的次数
- 历史记录字典：用于存储学习过程数据

choose_action　根据 UCB 规则选择价格曲线：

1. 第一轮强制选择免费价格曲线（索引 0）
2. 计算经验估计：$\bar{q}_i = \frac{N_i}{T_i}$

3. 计算置信度奖励：$\sqrt{\frac{\log T}{T_i}}$

4. 构造乐观估计：$\hat{q}_i = \bar{q}_i + \sqrt{\frac{\log T}{T_i}}$

5. 计算 UCB 收入：$\hat{rev}(p) = \sum_{i=1}^{m} \hat{q}_i \cdot revenue(p,i)$

6. 选择 UCB 收入最高的价格曲线

update　根据环境反馈更新内部状态：

- 更新探索次数：$T_i \leftarrow T_i + 1$（对于所有会购买类型 i 的买家）
- 更新观察次数：$N_i \leftarrow N_i + 1$（如果本轮观察到类型 i 买家）
- 记录历史数据：回合数、动作、收入、买家类型等

3.1.4　visualization.py

最后根据记录到的历史数据，绘制出 UCB 算法的可视化表现，代码不再赘述，接下来展示结果

3.1.5 实验结果

图 3.1: UCB 算法结果

可以看到，学习到的估计和真实分布非常接近，说明 UCB 算法能够很好地学习到真实分布。

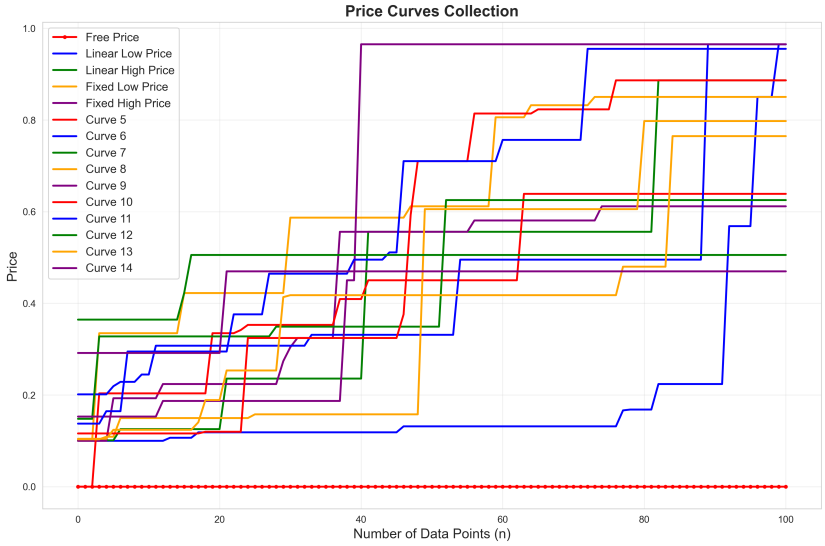

图 3.2: 价格曲线

可以看到价格曲线确实是阶梯状的，符合论文中的描述。

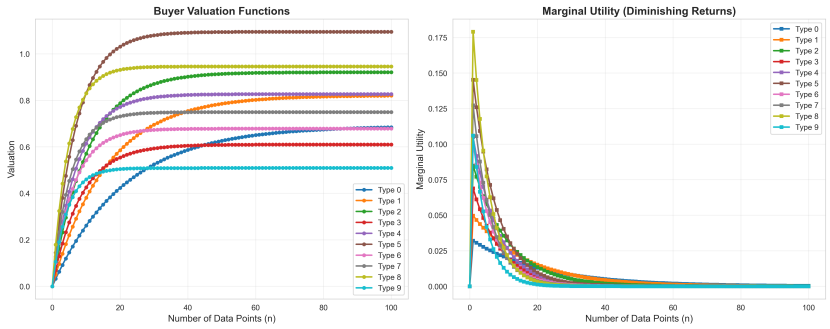

图 3.3: 估值函数

估值函数也与满足我们的要求

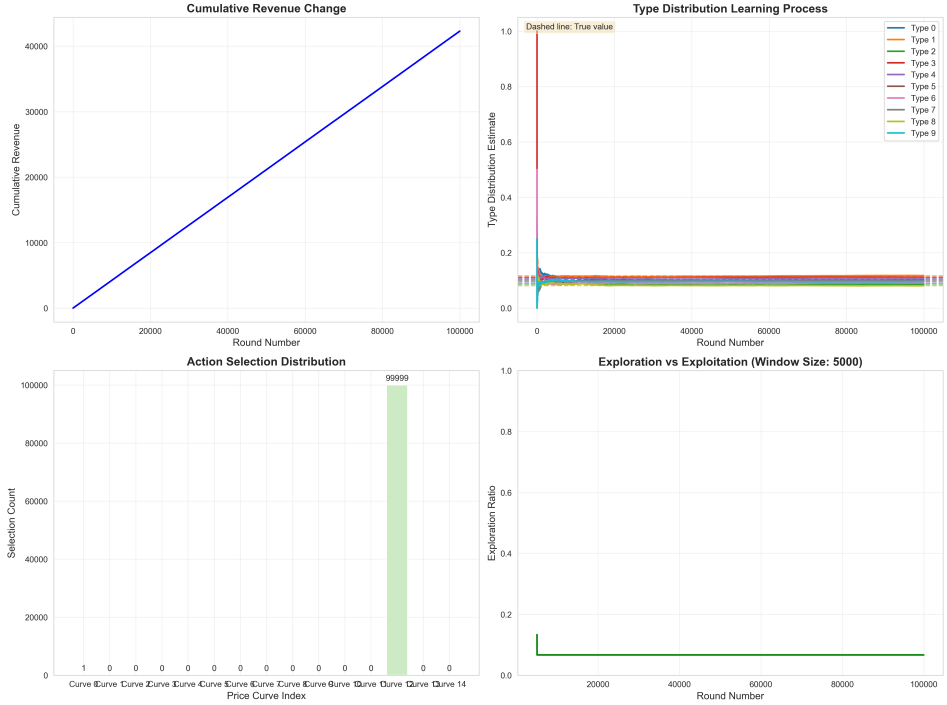

图 3.4: 学习过程

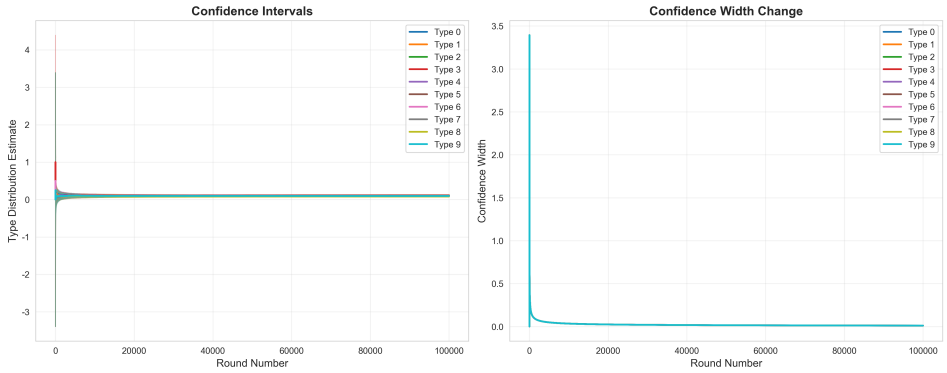

图 3.5: UCB 置信界

随着回合数的增加，置信界逐渐缩小，也满足我们的预期；

3.2 实验部分 - FTPL

3.2.1 基本结构

本实验主要分为两部分：

- FTPL.py: 主程序，实现 FTPL 算法

- utils.py: 工具函数，负责处理数据（离散化）

3.2.2 utils.py

离散化价格空间

首先需要离散化价格空间，将连续的价格区间 $[0,1]$ 离散化为点集。

- 设置 Z_i 点作为分割区间的依据：$Z_i = \epsilon(1+\epsilon)^i, \quad i \in \left\{0,1,\ldots,\left\lceil \log_{1+\epsilon}\frac{1}{\epsilon}\right\rceil\right\}$
- 根据 Z_i 点，构建离散化价格空间 W：
  - 区间 W_i 是范围为 $[Z_{i-1}, Z_{i+1})$ 的均匀离散化，间隔为 $Z_{i-1}\cdot\frac{\epsilon}{m}$
  - 更具体的，$W_i = \{Z_{i-1} + Z_{i-1}\cdot\frac{\epsilon k}{m} \mid k \in \{1,2,\ldots,\lceil(2+\epsilon)m\rceil\}\}$
  - 将这些区间并起来，我们就得到了离散化的价格空间 $W = \bigcup_{i=1}^{\lceil\log_{1+\epsilon}\frac{1}{\epsilon}\rceil} W_i$

现在我们可以初步构建离散的价格曲线空间 $\overline{\mathcal{P}}$，其中每个价格曲线 p 是一个函数，定义为 $p:[N]\to W$。这个价格曲线空间的复杂度仍然有些高，因此我们需要更进一步，将数量空间离散化

离散化物品数量空间

物品数量空间为 $[0,N]$，虽然其已经是离散的，但为了进一步减少其复杂度，我们将其离散化为更小的点集 N_D。

- 定义阈值 $\lfloor\frac{2Jm}{\epsilon^2}\rfloor$，我们只离散化高于这个阈值的数量
- 设置 Y_i 点作为分割区间的依据：$Y_i = \lfloor\frac{2Jm}{\epsilon^2}(1+\epsilon^2)^i\rfloor, \quad i = 0,\ldots,\lceil\log_{1+\epsilon^2}\frac{N\epsilon^2}{2Jm}\rceil$
- 对于每个 Y_i，我们在区间 $[Y_i, Y_{i+1})$ 上均匀离散化，间隔为 $Y_i\cdot\frac{\epsilon^2}{2Jm}$，得到点集 Q_i
- 将这些区间并起来，得到：$Q = \bigcup_{i=0}^{\lceil\log_{1+\epsilon^2}\frac{N\epsilon^2}{2Jm}\rceil} Q_i$
- 最后补上阈值及之下的点集，得到 $N_D = \{1,2,\ldots,\lfloor\frac{2Jm}{\epsilon^2}\rfloor\}\cup Q$

构建阶梯价格曲线

现在我们可以构建阶梯价格曲线集合 $\overline{\mathcal{P}}$，其中每个价格曲线 p 是一个函数，定义为 $p : N_D \to W$。

- 由于可能的价格曲线非常多，这里使用了采样数量 'sample_num' 来作为随机采样的数量
- 每轮采样，先从 1 到 m 中随机选择一个整数 k，作为阶梯的数量
- 如果 $k = 1$，那么函数是常数函数，直接从价格空间中随机选择一个价格，构建从 $[N]$ 到该价格的函数
- 如果 $k > 1$，则要构建一个多段的阶梯函数，这个阶梯函数有 $k-1$ 个跳变点和 k 个价格点
  - 首先选择 $k-1$ 个跳变点 $j_1, j_2, \ldots, j_{k-1}$，这些点从数量空间 N_D 中随机采样，排除 0 和 N
  - 然后选择 k 个价格点 $p_1, p_2, \ldots, p_k$，这些点从价格空间 W 中随机采样
  - 最后将这些价格点和跳变点组合成一个阶梯函数得到价格曲线 p 如下：

$$p(n) = \begin{cases} p_1 & \text{if } 0 \le n < j_1 \\ p_2 & \text{if } j_1 \le n < j_2 \\ \vdots & \vdots \\ p_k & \text{if } j_{k-1} \le n \le N \end{cases}$$

- 这样就得到了一个阶梯价格曲线集合 $\overline{\mathcal{P}}$

3.2.3 FTPL.py

基本参数

- N: 物品数量（默认 100）
- m: 买家类型数量，同时也是阶梯函数的最大阶数（默认 20）
- J: 收益递减常数（默认 2）
- T: 总回合数（默认 20）
- ϵ: 近似精度参数（默认 0.08）

- θ: 随机噪声参数（默认 1）
- $sample\_num$: 每轮采样的价格曲线数量（默认 20000）
- $valuation$: 买家估值函数，定义为 $v_i(n) = V - b \cdot e^{-a \cdot n}$
  - V: 基础价值，随着买家类型变化（例如 $0.2 + \frac{buyer\_type}{T}$）
  - a: 递减率，随着买家类型变化（例如 $0.1 + buyer\_type \cdot 0.01$）
  - b: 价格影响系数，随着买家类型变化（例如 $0.2 + buyer\_type \cdot 0.01$）

FTPL 算法

FTPL 部分的步骤如下：

- 初始化参数和变量，包括价格曲线集合 $\overline{\mathcal{P}}$ 和以及每个价格曲线的收益
- 在每轮中，执行如下步骤：
  1. 卖家根据策略选择一个价格曲线 $p \in \overline{\mathcal{P}}$
  2. 一个对抗性策略选择买家类型 $t \in [m]$，以降低卖家的收益
  3. 买家根据自己类型进行最优购买
  4. 计算卖家的收益 $r_t(p)$，同时更新并记录所有信息
- 重复上述步骤直到达到总回合数 T，最终会将记录的信息可视化展示

卖家定价

- 在开始总买卖之前，先对每个价格曲线 $p \in \overline{\mathcal{P}}$ 从分布 $\theta e^{-\theta x}$ 中采样一个噪声 θ_p
- 第 t 轮中，计算每条价格曲线 p 的历史收益（含噪声）$r(p) = \sum_{i=1}^{t-1} r_i(p) + \theta_p$
- 选择收益最大的价格曲线 $p_t = \arg\max_{p \in \overline{\mathcal{P}}} r(p)$

买家最优购买

- 买家类型 i 根据价格曲线 p_t 和估值函数 $v_i(n)$ 进行购买决策
- 对于每种购买数量，计算效用函数 $u_i(n) = v_i(n) - p_t(n)$
- 如果所有效用不大于 0，则不购买，否则选择效用最大的购买量 $n_{i_t,p_t} = \arg\max_n u_i(n)$
- 如果存在多个效用相等的最大值，则选择最大的 n

对抗性生成买家类型

- 对抗性策略会选择一个买家类型 i_t，使得卖家的收益最小化
- 具体来说，对于每个买家类型 i，由最优购买策略计算当前价格曲线下的购买数量 n_{i,p_t}
- 选择使得卖家收益 $r_t(p_t)$ 最小的买家类型 $i_t = \arg\min_i p(n_{i_t,p_t})$

收益计算与更新

- 根据买家购买数量 n_{i_t,p_t}，来分类计算卖家的收益
- 如果 $n_{i_t,p_t} > 0$，则对所有价格曲线 $p \in \overline{\mathcal{P}}$，其本轮收益为 $r_t(p) = p(n_{i_t,p})$
- 如果 $n_{i_t,p_t} = 0$，则：
  - 将所有在当前价格曲线下购买数量为 0 的买家记为集合 S_t^c
  - 对所有价格曲线 $p \in \overline{\mathcal{P}}$，其本轮收益为 $r_t(p) = \sum_{i \in S_t^c} p(n_{i,p})$
- 计算遗憾值：$R_t = \max_{p \in \overline{\mathcal{P}}} \sum_{t=1}^{T} p(n_{i_t,p}) - \sum_{t=1}^{T} p_t(n_{i_t,p_t})$，即卖家在最好的价格曲线下的总收益减去实际的总收益
- 记录历史收益、遗憾值以及买家类型等信息

3.2.4 结果分析

```
买家类型序列： [0, 1, 10, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0]
每轮购买数量： [18, 0, 0, 38, 0, 38, 38, 38, 38, 38, 38, 38, 38, 0, 38, 38, 38, 38, 38, 38]
每轮支付金额： [0.11 0.   0.   0.5  0.   0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.
 0.5  0.5  0.5  0.5  0.5  0.5 ]
每轮累计收益： [0.11 0.11 0.11 0.61 0.61 1.11 1.61 2.11 2.61 3.11 3.61 4.11 4.61 4.61
 5.11 5.61 6.11 6.61 7.11 7.61]
每轮的历史奖励总和： [8.66 5.33 4.59 ... 5.44 4.36 4.06]
遗憾： [0.88 1.78 1.78 2.23 2.23 2.68 3.13 3.58 4.03 4.48 4.93 5.38 5.83 5.83
 6.28 6.73 7.18 7.63 8.08 8.53]
```

图 3.6: FTPL 终端输出

这里展示了 FTPL 算法的终端输出结果，包括每轮的收益、遗憾值以及买家类型等信息，具体可视化结果在后面的图表中。

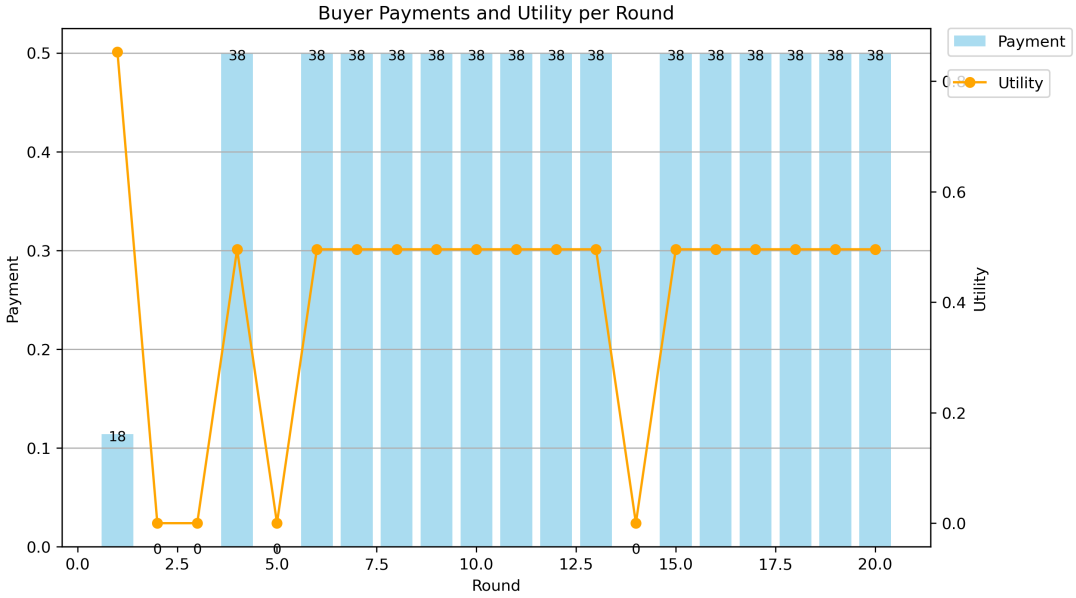

图 3.7: FTPL 买家支付及效益情况

柱状图展示了每轮中卖家的支付情况，左侧坐标为买家支付的金额，右侧坐标为买家的效用值。柱状图上方的数字为购买的数量。叠加的折线图展示了买家的效用情况。

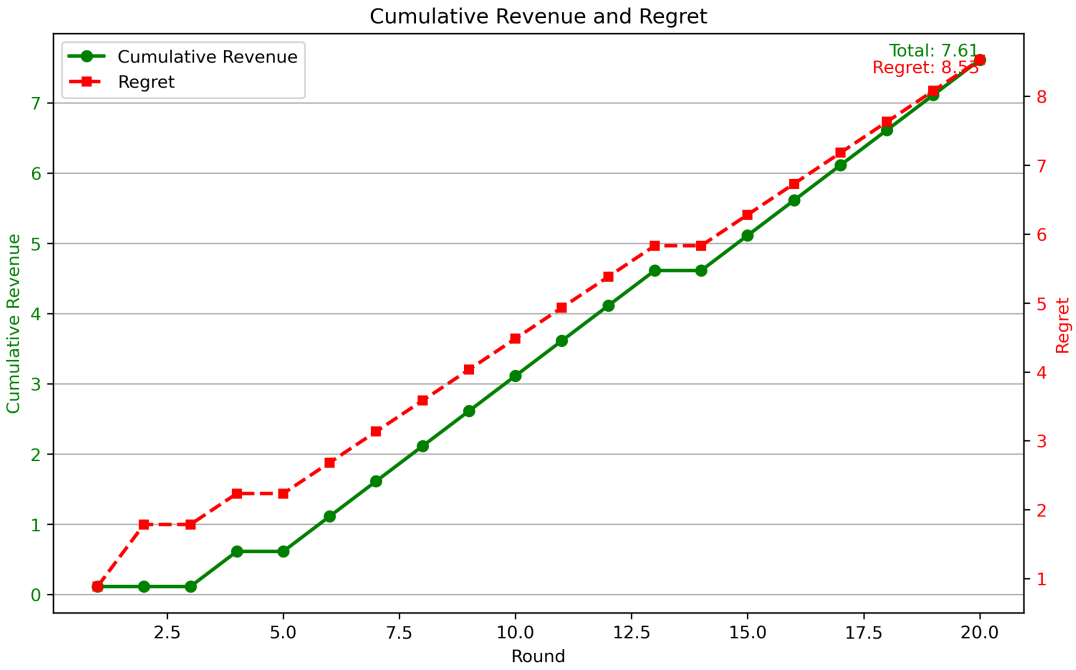

图 3.8: FTPL 累计收益及遗憾

绿色实现表示卖家的累计收益，红色虚线表示卖家的遗憾值。随着轮次的增加，

卖家的累计收益和遗憾值逐渐增加。

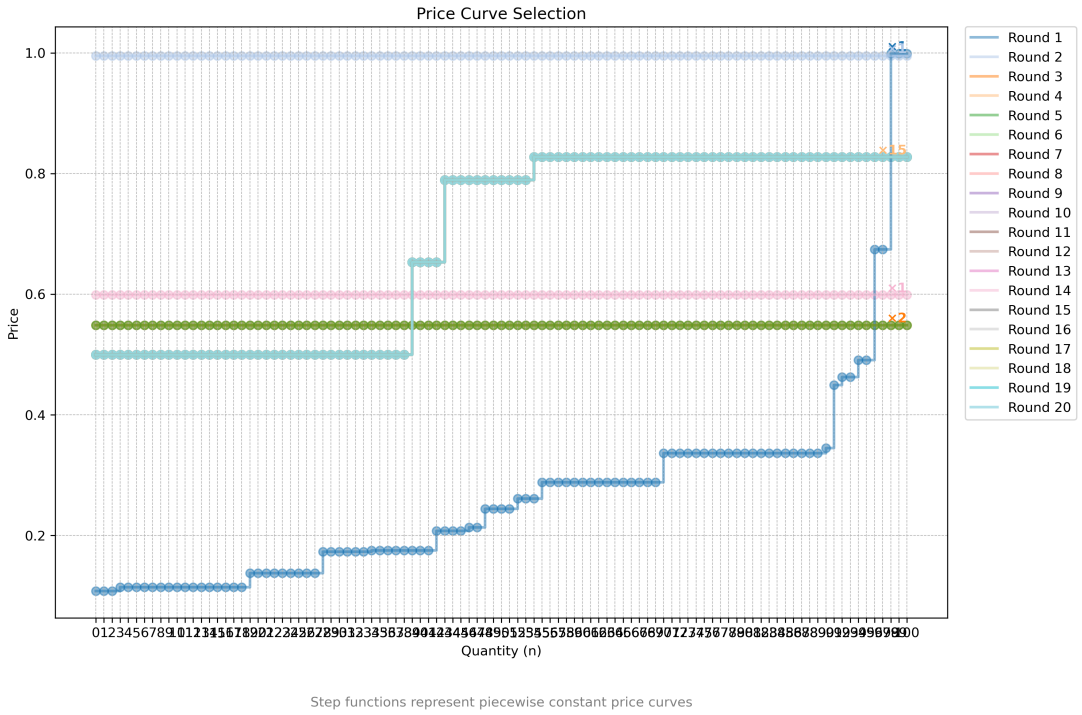

图 3.9: FTPL 价格曲线选择情况

这里使用了折线图展示了所有的价格曲线的选择情况。横坐标为轮次，纵坐标为价格，标记了不同价格曲线的选择次数。

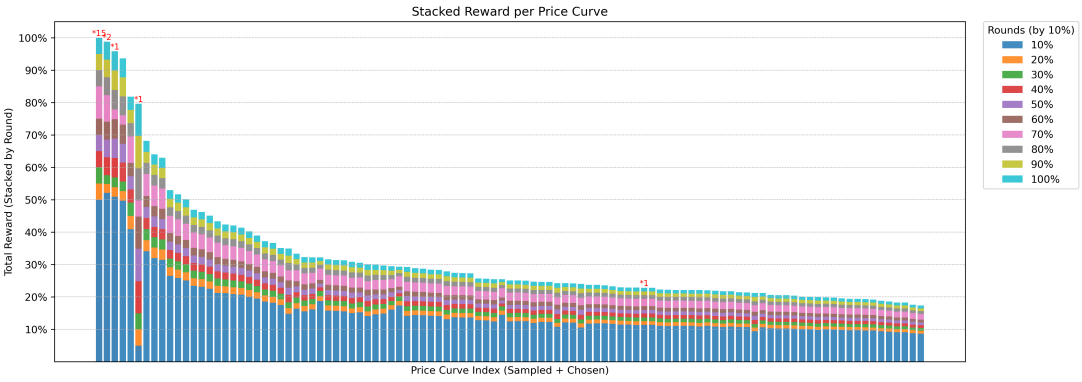

图 3.10: FTPL 奖励分布

这里展示了 FTPL 算法中每个价格曲线的奖励分布情况。横坐标为价格曲线的索引，纵坐标为对应的奖励值。为了方便显示，这里从所有价格曲线中随机选择了 100 条，并加上所有被选中的价格曲线进行展示；纵向按照轮数的 10% 进行划分和累加。价格曲线按照总奖励值从大到小排序，用选中次数来标记被选中的价格曲线。

可以看到被选中的价格曲线的奖励都比较高，这正反映了 FTPL 算法的选择策略。

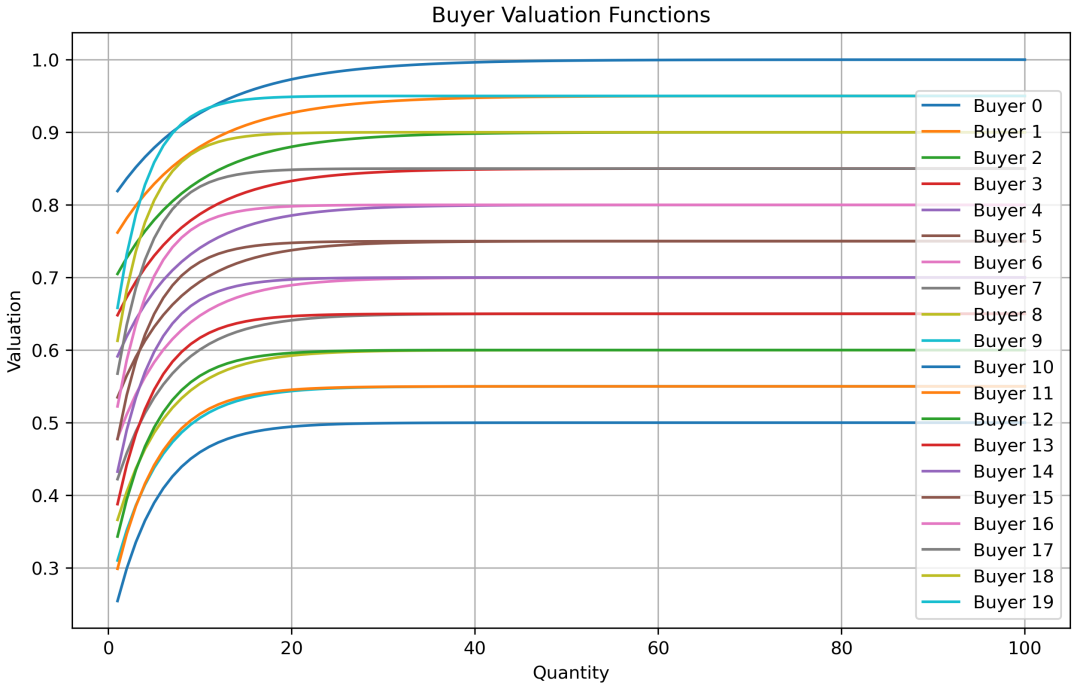

图 3.11: FTPL 买家估值函数

这里展示了 FTPL 算法中买家的估值函数。横坐标为物品数量，纵坐标为估值函数的值，不同颜色的曲线表示不同买家类型的估值函数。