在线学习与数据定价

2024-2025 学年短学期 数据要素市场

吴一航

yhwu_is@zju.edu.cn

浙江大学计算机科学与技术学院

2024 年 7 月 9 日

机器学习

机器学习（machine learning）利用算法，无需人工交互即可完成学习

- 学习：从输入的数据中通过算法的运行分析和解释数据，学习到数据的模式及结构，从而做出更好的决策等
- 机器学习通常有如下类别：
  - 监督学习、无监督学习、半监督学习、主动学习
  - 强化学习、在线学习
  - 深度学习
  - …

在线算法

“在线”这个词我们并不陌生，在算法课程中我们已经接触过了在线算法，其大致流程如下：[1]

Online Algorithm Template

On an instance I, including an ordering of the data items $(x_1, \ldots, x_n)$:
$i := 1$
While there are unprocessed data items
    The algorithm receives $x_i \in R$ and makes an irrevocable decision $d_i \in A$ for x_i (based on x_i and all previously seen data items and decisions).
    $i := i + 1$
EndWhile

- 在线算法中，输入是一个序列．在每个输入项进入算法之后，算法需要输出一个决策，并且该决策是不可反悔的
- 目标是最大化或最小化目标函数，使用**竞争比（competitive ratio）**来衡量算法的性能，即在线算法解与最优离线算法解的比值
- 例如在线装箱问题，paging 问题等

[1] 本次课程主要参考 UChicago CMSC 35401 相关章节

在线学习与在线算法

在线学习中的“在线”含义类似，都是指在数据流到来的过程中，算法需要实时地做出决策，但是“学习”具有更深层次的含义

- 在线学习中我们可以逐步探索最优的反应是什么，而在线算法，例如装箱问题，我们通常按照固定的装箱规则进行装箱

在线学习与在线算法

在线学习中的“在线”含义类似，都是指在数据流到来的过程中，算法需要实时地做出决策，但是“学习”具有更深层次的含义

- 在线学习中我们可以逐步探索最优的反应是什么，而在线算法，例如装箱问题，我们通常按照固定的装箱规则进行装箱

下面我们通过一个定价的例子领悟在线学习的含义

例

假设你要出售一份数据，你知道会有 N 个人来购买你的数据，并且每个人对数据的估值 v 都完全一致，都在 $[0,1]$ 中．买家是逐个到达的，你需要提供一个价格 p，如果 $v \geqslant p$，买家就会购买你的数据，否则买家会离开．你的目标是尽快地学习到 v 的值，误差范围是 $\varepsilon = 1/N$.

在线学习与在线算法

在线学习中的“在线”含义类似，都是指在数据流到来的过程中，算法需要实时地做出决策，但是“学习”具有更深层次的含义

- 在线学习中我们可以逐步探索最优的反应是什么，而在线算法，例如装箱问题，我们通常按照固定的装箱规则进行装箱

下面我们通过一个定价的例子领悟在线学习的含义

例

假设你要出售一份数据，你知道会有 N 个人来购买你的数据，并且每个人对数据的估值 v 都完全一致，都在 $[0,1]$ 中．买家是逐个到达的，你需要提供一个价格 p，如果 $v \geqslant p$，买家就会购买你的数据，否则买家会离开．你的目标是尽快地学习到 v 的值，误差范围是 $\varepsilon = 1/N$.

一个最简单的方法：在出售前询问买家的估值，但是很可能出现欺骗行为

二分搜索

一个最 naive 的方法就是二分搜索，我们最多需要 $\log N$ 次的搜索来实现这一精度.

- 如果的确遇到了最坏的情况，$\log N$ 次搜索之后，我们可以设置到价格 $p \geqslant \tilde{v} - 1/N$，其中 $\tilde{v}$ 是我们第 $\log N$ 轮学习到的值．在这种情况下，N 轮之后，我们的总收益是

$$\underbrace{0}_{\text{前 } \log N \text{ 轮}} + \underbrace{(N - \log N)(v - 2/N)}_{\text{后} N - \log N \text{ 轮}} \approx vN - v \log N - 2$$

二分搜索

一个最 naive 的方法就是二分搜索，我们最多需要 $\log N$ 次的搜索来实现这一精度.

- 如果的确遇到了最坏的情况，$\log N$ 次搜索之后，我们可以设置到价格 $p \geqslant \tilde{v} - 1/N$，其中 $\tilde{v}$ 是我们第 $\log N$ 轮学习到的值. 在这种情况下，N 轮之后，我们的总收益是

$$\underbrace{0}_{\text{前 } \log N \text{ 轮}} + \underbrace{(N - \log N)(v - 2/N)}_{\text{后} N - \log N \text{ 轮}} \approx vN - v \log N - 2$$

为了衡量二分搜索的性能，我们引入**遗憾（regret）**的概念

定义

遗憾是指我们的总收益与最优总收益之间的差值，即和在开始出售前已经知道准确的 v 的情况的差值.

二分搜索的遗憾

我们可以计算出二分搜索的遗憾

$$\text{遗憾} \approx vN - (vN - v\log N - 2) = v\log N + 2$$

二分搜索的遗憾

我们可以计算出二分搜索的遗憾

$$\text{遗憾} \approx vN - (vN - v\log N - 2) = v\log N + 2$$

- 这是最小的可能遗憾吗?

二分搜索的遗憾

我们可以计算出二分搜索的遗憾

$$\text{遗憾} \approx vN - (vN - v\log N - 2) = v\log N + 2$$

- 这是最小的可能遗憾吗?

定理 (Kleinberg/Leighton, FOCS'03)

存在一个算法，使得其遗憾至多为 $1 + 2\log\log N$.

二分搜索的遗憾

我们可以计算出二分搜索的遗憾

$$\text{遗憾} \approx vN - (vN - v\log N - 2) = v\log N + 2$$

- 这是最小的可能遗憾吗?

定理 (Kleinberg/Leighton, FOCS'03)

存在一个算法，使得其遗憾至多为 $1 + 2\log\log N$.

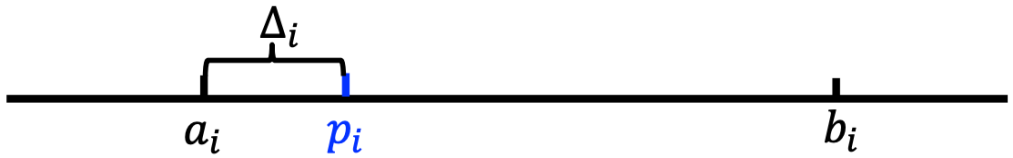
为什么二分搜索还有改进的空间?

- 尽管这是我们在没有任何预知消息的情况下能搜索到 N 的最快算法，但当猜测的 $p_i > v$ 时，我们一分钱也赚不到，因此二分搜索在向上探索的时候可能过于激进

因此我们改进的算法需要在探索时更加保守!

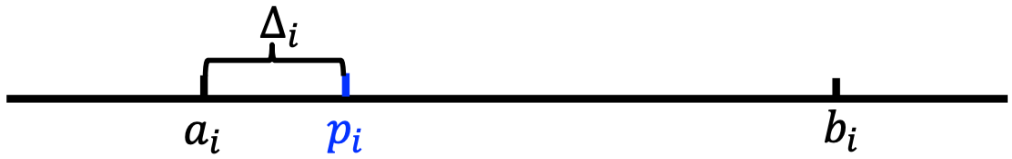
改进的算法

1. 在第 i 阶段，我们保持有一个探索区间 $[a_i, b_i]$，其中 $a_1 = 0, b_1 = 1$，并设置 $\Delta_1 = 1/2$
2. 为第 i 个买家 i 提供价格 $p_i = a_i + \Delta_i$

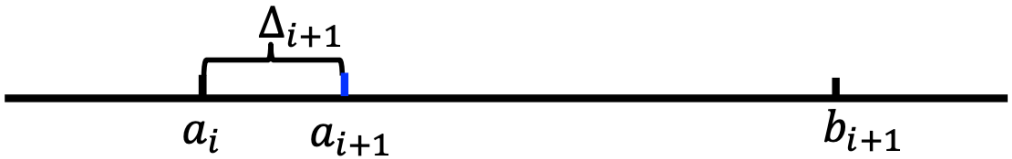


改进的算法

1. 在第 i 阶段，我们保持有一个探索区间 $[a_i, b_i]$，其中 $a_1 = 0, b_1 = 1$，并设置 $\Delta_1 = 1/2$
2. 为第 i 个买家 i 提供价格 $p_i = a_i + \Delta_i$

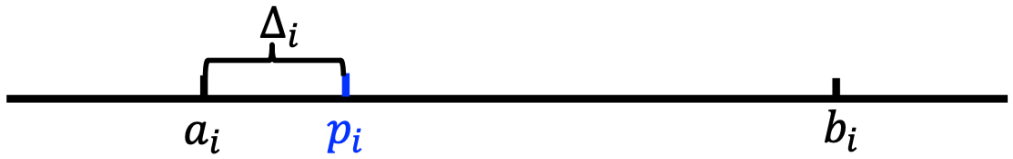


3. 如果 i 接受价格 p_i，则 $a_{i+1} = p_i, b_{i+1} = b_i, \Delta_{i+1} = \Delta_i$；

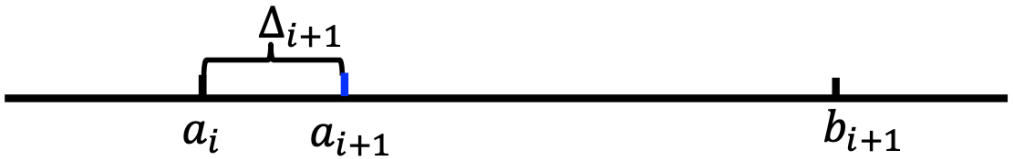


改进的算法

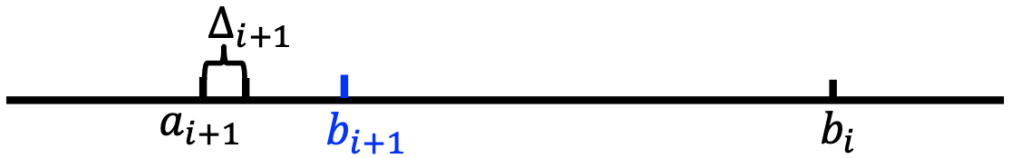
1. 在第 i 阶段，我们保持有一个探索区间 $[a_i, b_i]$，其中 $a_1 = 0, b_1 = 1$，并设置 $\Delta_1 = 1/2$
2. 为第 i 个买家 i 提供价格 $p_i = a_i + \Delta_i$



3. 如果 i 接受价格 p_i，则 $a_{i+1} = p_i, b_{i+1} = b_i, \Delta_{i+1} = \Delta_i$；

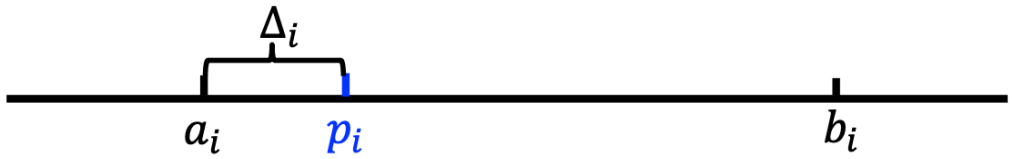


4. 如果 i 拒绝价格 p_i，则 $a_{i+1} = a_i, b_{i+1} = p_i, \Delta_{i+1} = (\Delta_i)^2$

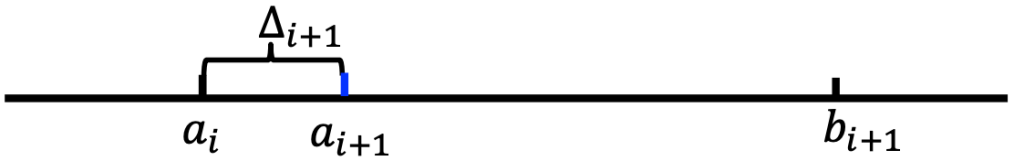


改进的算法

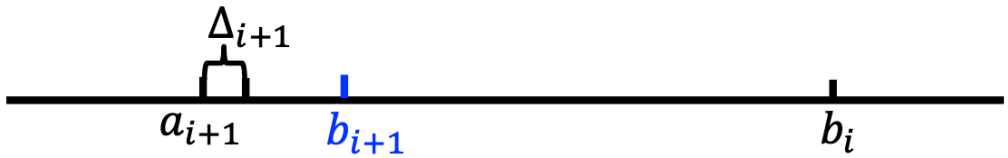
1. 在第 i 阶段，我们保持有一个探索区间 $[a_i, b_i]$，其中 $a_1 = 0, b_1 = 1$，并设置 $\Delta_1 = 1/2$
2. 为第 i 个买家 i 提供价格 $p_i = a_i + \Delta_i$



3. 如果 i 接受价格 p_i，则 $a_{i+1} = p_i, b_{i+1} = b_i, \Delta_{i+1} = \Delta_i$；



4. 如果 i 拒绝价格 p_i，则 $a_{i+1} = a_i, b_{i+1} = p_i, \Delta_{i+1} = (\Delta_i)^2$



5. 当 $b_i - a_i \leqslant 1/N$ 时，此后价格均取 a_i，不再改变

改进的算法分析

定理

改进的算法的遗憾至多为 $1+2\log\log N$.

改进的算法分析

定理

改进的算法的遗憾至多为 $1+2\log\log N$.

首先我们需要分析 b_i-a_i 的特点，我们有如下结论：

引理 1

$\Delta_i=2^{-2^{i-1}}$，并且当 $\Delta_{i+1}=(\Delta_i)^2$ 时，$b_{i+1}-a_{i+1}=\Delta_i=\sqrt{\Delta_{i+1}}$.

改进的算法分析

定理

改进的算法的遗憾至多为 $1+2\log\log N$.

首先我们需要分析 b_i-a_i 的特点，我们有如下结论：

引理 1

$\Delta_i=2^{-2^{i-1}}$，并且当 $\Delta_{i+1}=(\Delta_i)^2$ 时，$b_{i+1}-a_{i+1}=\Delta_i=\sqrt{\Delta_{i+1}}$.

证明：第一个结论根据数学归纳法可以证明：

1. 当 $i=1$ 时，$\Delta_1=1/2=2^{-2^0}$
2. 假设对于 $i=k$ 成立，即 $\Delta_k=2^{-2^{k-1}}$，则

$$\Delta_{k+1}=(\Delta_k)^2=2^{-2^{k-1}-2^{k-1}}=2^{-2^k}$$

第二个结论，当 $\Delta_{i+1}=(\Delta_i)^2$ 时，根据算法直接得到 $b_i-a_i=\Delta_i=\sqrt{\Delta_{i+1}}$.

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1
2. 因此重点在于分析达到这一步之前的遗憾

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1
2. 因此重点在于分析达到这一步之前的遗憾
3. 在达到这一步之前 Δ 更新了多少次?
   1. $\log\log N$：令 $2^{-2^i} = 1/N$，则 $i = \log\log N$
   2. 接下来就要证明每个 Δ_i 内产生的遗憾是有限的

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1
2. 因此重点在于分析达到这一步之前的遗憾
3. 在达到这一步之前 Δ 更新了多少次?
   1. $\log\log N$：令 $2^{-2^i} = 1/N$，则 $i = \log\log N$
   2. 接下来就要证明每个 Δ_i 内产生的遗憾是有限的

引理 2

任意的步长 Δ_i 内的遗憾至多为 2.

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1
2. 因此重点在于分析达到这一步之前的遗憾
3. 在达到这一步之前 Δ 更新了多少次？
   1. $\log\log N$：令 $2^{-2^i} = 1/N$，则 $i = \log\log N$
   2. 接下来就要证明每个 Δ_i 内产生的遗憾是有限的

引理 2

任意的步长 Δ_i 内的遗憾至多为 2.

1. 如果在 Δ_i 下直接被拒绝，遗憾为 1
2. 如果发生出售，则最多出售 $\sqrt{\Delta_i}/\Delta_i$ 次，因为 $\Delta_{i-1} = \sqrt{\Delta_i}$，如果超出这个次数，那么 Δ_{i-1} 在前面的步骤不会更新．因此出售过程中的遗憾最多为

$$\frac{\sqrt{\Delta_i}}{\Delta_i} \times \sqrt{\Delta_i} = 1$$

改进的算法分析（Cont'd）

1. 在 $b_i - a_i \leqslant 1/N$ 后，总的遗憾最多为 1
2. 因此重点在于分析达到这一步之前的遗憾
3. 在达到这一步之前 Δ 更新了多少次？
   1. $\log\log N$：令 $2^{-2^i} = 1/N$，则 $i = \log\log N$
   2. 接下来就要证明每个 Δ_i 内产生的遗憾是有限的

引理 2

任意的步长 Δ_i 内的遗憾至多为 2.

1. 如果在 Δ_i 下直接被拒绝，遗憾为 1
2. 如果发生出售，则最多出售 $\sqrt{\Delta_i}/\Delta_i$ 次，因为 $\Delta_{i-1} = \sqrt{\Delta_i}$，如果超出这个次数，那么 Δ_{i-1} 在前面的步骤不会更新．因此出售过程中的遗憾最多为

$$\frac{\sqrt{\Delta_i}}{\Delta_i} \times \sqrt{\Delta_i} = 1$$

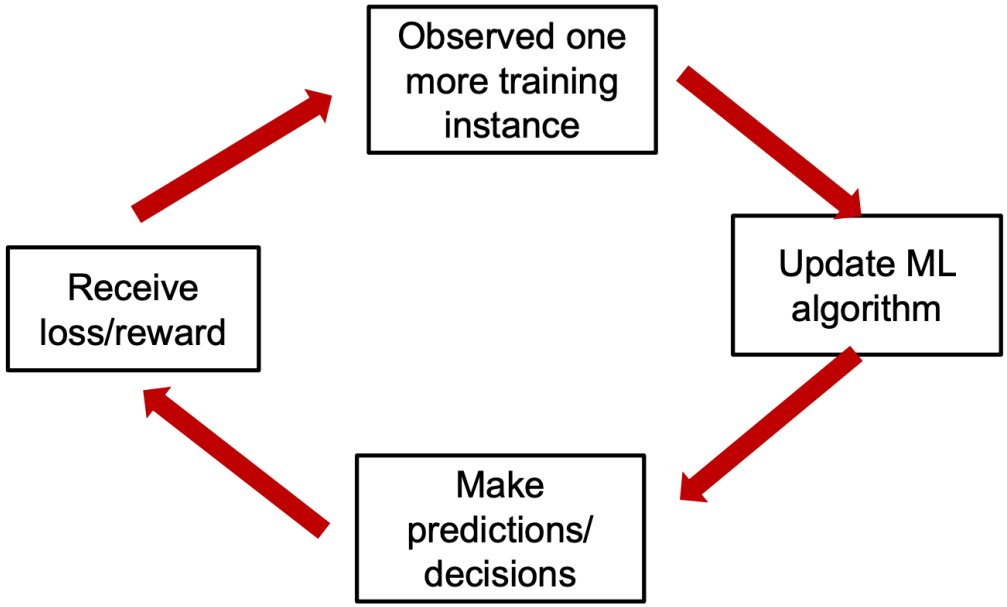
综合引理 1 和引理 2，我们可以得到改进算法的遗憾为 $1 + 2\log\log N$.

例子的评注

1. 这个例子展现了在线学习中**探索（exploration）**和**利用（exploitation）**的平衡
   - 探索指的是我们想要学习到 v 的值
   - 但我们最终的目标是学到 v 来最大化利润
   - 推荐系统的例子
2. 启示：二分搜索用了更快的探索路径，但是在探索的过程中可能会有更大的遗憾
3. 模型的拓展
   - 如果同一个买家会重复出现呢？可能会扰乱局势？
   - 如果不止出售一个物品呢？

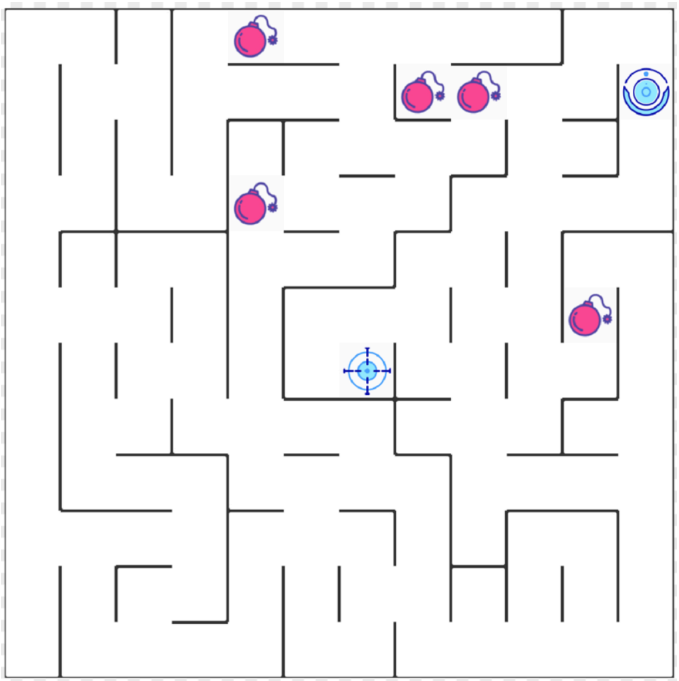
在线学习的流程

刚刚的我们提供了一个在线学习的基本例子，下面我们可以总结一下在线学习的基本流程



在线学习的应用

1. 玉泉往返紫金港的交通方式
   - 地铁？公交车？走路？出租车？电动车？自行车？
   - 人实际上就是在不断的在线学习
2. 强化学习与股票投资、广告拍卖的投标
3. 推荐系统
4. 机器人
5. 医疗诊断与治疗
6. …



在线学习基本模型

1. 每个学习者在一个不确定的世界中，要做一个 T 步的决策序列
2. 在每一步 $t = 1, 2, \ldots, T$ 中
   1. 学习者做出一个行动
   2. 这一行动可能带来代价（cost），也可能带来奖励（reward）
   3. 观察到世界的一些反馈（feedback）
   4. 最后根据行动结果和反馈来调整自己的行动
3. 学习者的目标：最小化总代价 / 最大化总收益

在线学习基本模型

1. 每个学习者在一个不确定的世界中，要做一个 T 步的决策序列
2. 在每一步 $t = 1, 2, \ldots, T$ 中
   1. 学习者做出一个行动
   2. 这一行动可能带来代价（cost），也可能带来奖励（reward）
   3. 观察到世界的一些反馈（feedback）
   4. 最后根据行动结果和反馈来调整自己的行动
3. 学习者的目标：最小化总代价 / 最大化总收益
4. 有两个重要的假定会影响模型的解法
   1. 代价 / 奖励：是从固定的分布中产生的还是恶意对抗产生的
   2. 反馈：只能看到自己行动的结果，还是部分情况的结果，还是完全的结果

本次课：代价是恶意生成的（有一个对手），反馈是完全的结果（例如股票市场）

在线学习基本模型（Cont'd）

关于反馈是否是完全结果，我们有非常多的现实例子可以参考

1. 反馈完全
   1. 股票市场
2. 反馈不完全
   1. 多臂老虎机
   2. 在线广告位
   3. 推荐系统
   4. 治疗疾病
   5. 扑克牌游戏、市场竞争等不完全信息博弈

在线学习形式化模型

在每一步 $t = 1, 2, \ldots, T$ 中：

1. 学习者选择一个行动集合 $[n] = \{1, \cdots, n\}$ 上的概率分布 p_t；
2. 对手在已知 p_t 的情况下选择一个代价向量 $c_t \in [0, 1]^n$，即为每一个行动选择一个代价；
3. 学习者选择一个行动 $i_t \sim p_t$，并且观察到代价 $c_t(i_t)$；
4. 学习者学到整个 c_t 以调整未来的策略

在线学习形式化模型

在每一步 $t = 1, 2, \ldots, T$ 中：

1. 学习者选择一个行动集合 $[n] = \{1, \cdots, n\}$ 上的概率分布 p_t；
2. 对手在已知 p_t 的情况下选择一个代价向量 $c_t \in [0, 1]^n$，即为每一个行动选择一个代价；
3. 学习者选择一个行动 $i_t \sim p_t$，并且观察到代价 $c_t(i_t)$；
4. 学习者学到整个 c_t 以调整未来的策略

注：

1. 学习者的目标：选取一个策略序列 $p_1, p_2, \ldots, p_T$ 使得总代价最小，即最小化期望代价 $\mathbf{E}_{i_t \sim p_t}\left[\sum_{t=1}^{T} c_t(i_t)\right]$
2. 对手（adversary）不一定非要存在，其实只是一个最差结果分析

在线学习算法的评价标准

接下来我们希望找到一个“好的”在线学习算法，就像前面定价的例子一样．我们的想法仍然是定义遗憾，但此时遗憾的定义不能太简单：

例

设行动集合为 $\{1,2\}$，在每一轮 t，对手按如下步骤选择代价向量：假设算法选择一个概率分布 p_t，如果在此分布下选择行动 1 的概率至少为 $1/2$，那么 $c_t=(1,0)$，反之 $c_t=(0,1)$．在此情况下，在线算法期望代价至少为 $T/2$，而在事先知晓代价向量的情况下，最优算法的期望代价为 0．

在线学习算法的评价标准

接下来我们希望找到一个“好的”在线学习算法，就像前面定价的例子一样．我们的想法仍然是定义遗憾，但此时遗憾的定义不能太简单：

例

设行动集合为 $\{1,2\}$，在每一轮 t，对手按如下步骤选择代价向量：假设算法选择一个概率分布 p_t，如果在此分布下选择行动 1 的概率至少为 $1/2$，那么 $c_t=(1,0)$，反之 $c_t=(0,1)$．在此情况下，在线算法期望代价至少为 $T/2$，而在事先知晓代价向量的情况下，最优算法的期望代价为 0．

这一例子表明，与最优离线算法比较可能出现线性级别的遗憾，因此这一基准太强了．因此我们转而将遗憾定义为在线算法与最优固定行动离线算法的代价之差．

在线学习算法的评价标准（Cont'd）

定义 (遗憾)

固定代价向量 $c_1, c_2, \ldots, c_T$，决策序列 $p_1, p_2, \ldots, p_T$ 的遗憾为

$$R_T = \mathbf{E}_{i_t \sim p_t}\left[\sum_{t=1}^{T} c_t(i_t)\right] - \min_{i \in [n]} \sum_{t=1}^{T} c_t(i)$$

在线学习算法的评价标准（Cont'd）

定义 (遗憾)

固定代价向量 $c_1, c_2, \ldots, c_T$，决策序列 $p_1, p_2, \ldots, p_T$ 的遗憾为

$$R_T = \mathbf{E}_{i_t \sim p_t}\left[\sum_{t=1}^{T} c_t(i_t)\right] - \min_{i \in [n]} \sum_{t=1}^{T} c_t(i)$$

即我们的遗憾被定义成了和最优的固定行动的代价之差，这样的定义相对而言更加合理

- 在前面的例子中，固定策略序列（全选 0 或 1）的遗憾不再是简单的 0
- 这一遗憾又称为“外部遗憾”，还有其它的遗憾（如交换遗憾）在后面介绍
- 平均遗憾：$\overline{R}_T/T$，若 $T \to \infty$，$\overline{R}_T/T \to 0$，则称算法是**无憾 (no-regret)** 的；等价的即 R_T 关于 T 是次线性的
- 这一定义的合理性在于，有自然的算法实现无憾，但无憾的实现也不是平凡的

跟风算法

接下来我们的目标就是设计一个无憾的在线学习算法．我们首先介绍一个最简单的在线学习算法：**跟风算法（Follow-The-Leader, FTL）**.

定义 (跟风算法)

跟风算法指在每一个时间点 t，选择最小累积代价 $\sum_{s=1}^{t-1} c_s(i)$ 的行动 i.

跟风算法

接下来我们的目标就是设计一个无憾的在线学习算法．我们首先介绍一个最简单的在线学习算法：**跟风算法（Follow-The-Leader, FTL）**.

定义 (跟风算法)

跟风算法指在每一个时间点 t，选择最小累积代价 $\sum_{s=1}^{t-1} c_s(i)$ 的行动 i.

下面这一含有两个行动 $\{1,2\}$ 的例子表明，跟风算法不是无憾的：

t	1	2	3	4	5	...	T
$c_t(1)$	1	0	1	0	1	...	*
$c_t(2)$	0	1	0	1	0	...	*

- FTL 会持续选择代价为 1 的行动，总代价为 T，而最优固定行动算法的代价为 $T/2$，平均遗憾为 $1/2$，故不是无憾算法

随机化是无憾的必要条件

事实上，任意的确定性算法都会有线性级别的遗憾：$\frac{n-1}{n}T$，其中 n 是可能行动的个数.

1. 对手是知道我们的算法的，因此如果采用确定性算法，对手可以直接推断出我们每一步的选择，从而设置我们选择的行为有代价 1，其余行为代价为 0
2. 这样学习者的总代价为 T，而最优固定行动算法的总代价最高为 T/n（因为有 n 种可能的行动）
3. 如果采用随机策略，则对手只能推断出我们的概率分布 p_t，而不能推断出我们的具体行动 $i_t \sim p_t$，因此对手无法完美利用我们的算法

随机化是无憾的必要条件

事实上，任意的确定性算法都会有线性级别的遗憾：$\frac{n-1}{n}T$，其中 n 是可能行动的个数.

1. 对手是知道我们的算法的，因此如果采用确定性算法，对手可以直接推断出我们每一步的选择，从而设置我们选择的行为有代价 1，其余行为代价为 0
2. 这样学习者的总代价为 T，而最优固定行动算法的总代价最高为 T/n（因为有 n 种可能的行动）
3. 如果采用随机策略，则对手只能推断出我们的概率分布 p_t，而不能推断出我们的具体行动 $i_t \sim p_t$，因此对手无法完美利用我们的算法

因此我们需要引入随机化来避免对手的推断，下面我们从一个简单的例子入手，引出我们的第一个无憾算法：乘性权重算法.

一个简单的例子

考虑一个简化的在线学习场景，每个行动的代价之可能为 0 或 1，并且存在一个完美的行动，其代价永远为 0（但学习者一开始不知道哪个是完美行动），是否存在次线性遗憾的算法?

一个简单的例子

考虑一个简化的在线学习场景，每个行动的代价之可能为 0 或 1，并且存在一个完美的行动，其代价永远为 0（但学习者一开始不知道哪个是完美行动），是否存在次线性遗憾的算法?

- 观察：只要一个行动出现了非零代价，那我们就可以永远排除它，但我们并不知道剩余的行动中哪个是最好的

一个简单的例子

考虑一个简化的在线学习场景，每个行动的代价之可能为 0 或 1，并且存在一个完美的行动，其代价永远为 0（但学习者一开始不知道哪个是完美行动），是否存在次线性遗憾的算法?

- 观察：只要一个行动出现了非零代价，那我们就可以永远排除它，但我们并不知道剩余的行动中哪个是最好的
- 因此我们的算法如下：对于每一步 $t = 1, 2, \ldots, T$，我们记录有截至目前从没出现过代价 1 的行动，然后在这些行动中根据均匀分布随机选择一个行动

算法分析

任一轮 t，令 $S_{good} = \{i \in [n] \mid$ 行动 i 从未出现过代价 $1\}$，$k = |S_{good}|$

- 因此每个 $i \in S_{good}$ 在下一轮被选中的概率为 $1/k$，而每个 $i \notin S_{good}$ 的概率为 0

算法分析

任一轮 t，令 $S_{good} = \{i \in [n] \mid$ 行动 i 从未出现过代价 $1\}$，$k = |S_{good}|$

- 因此每个 $i \in S_{good}$ 在下一轮被选中的概率为 $1/k$，而每个 $i \notin S_{good}$ 的概率为 0

对于任意的 $\varepsilon \in (0,1)$，下面两种情况之一一定会发生：

1. S_{good} 中至多 εk 个行动有代价 1，此时这一阶段的期望代价至多为 ε

算法分析

任一轮 t，令 $S_{good} = \{i \in [n] \mid \text{行动 } i \text{ 从未出现过代价 } 1\}$，$k = |S_{good}|$

- 因此每个 $i \in S_{good}$ 在下一轮被选中的概率为 $1/k$，而每个 $i \notin S_{good}$ 的概率为 0

对于任意的 $\varepsilon \in (0,1)$，下面两种情况之一一定会发生：

1. S_{good} 中至多 εk 个行动有代价 1，此时这一阶段的期望代价至多为 ε
2. S_{good} 中至少 εk 个行动有代价 1，每次出现这一情况时，下一步就可以排除掉至少 εk 个行动，因此这种情况最多出现 $\log_{1-\varepsilon} \frac{1}{n}$ 次

算法分析

任一轮 t，令 $S_{good} = \{i \in [n] \mid \text{行动 } i \text{ 从未出现过代价 } 1\}$，$k = |S_{good}|$

- 因此每个 $i \in S_{good}$ 在下一轮被选中的概率为 $1/k$，而每个 $i \notin S_{good}$ 的概率为 0

对于任意的 $\varepsilon \in (0,1)$，下面两种情况之一一定会发生：

1. S_{good} 中至多 εk 个行动有代价 1，此时这一阶段的期望代价至多为 ε
2. S_{good} 中至少 εk 个行动有代价 1，每次出现这一情况时，下一步就可以排除掉至少 εk 个行动，因此这种情况最多出现 $\log_{1-\varepsilon} \frac{1}{n}$ 次

因此总的遗憾至多为

$$R_T = T \times \varepsilon + \log_{1-\varepsilon} \frac{1}{n} \times 1 = T\varepsilon + \frac{\ln n}{-\ln(1-\varepsilon)} \leqslant T\varepsilon + \frac{\ln n}{\varepsilon}$$

最后的不等号来源于 $-\ln(1-\varepsilon) \geqslant \varepsilon (0 < \varepsilon < 1)$. 显然当 $\varepsilon = \sqrt{\frac{\ln n}{T}}$ 时，$R_T \leqslant 2\sqrt{T \ln n}$，即次线性遗憾.

更一般的情况

我们可以将之前的算法描述得更加形式化：

Initialize weight $w_1(i) = 1, \forall i = 1, \cdots n$
For $t = 1, \cdots, T$
1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in \{0,1\}^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - c_t(i))$

更一般的情况

我们可以将之前的算法描述得更加形式化：

Initialize weight $w_1(i) = 1, \forall i = 1, \cdots n$
For $t = 1, \cdots, T$
1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in \{0,1\}^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - c_t(i))$

在更一般的情况下，代价可以是 $[0, 1]$ 中的任意值，并且不一定存在完美策略，因此我们需要对算法进行修改：

Initialize weight $w_1(i) = 1, \forall i = 1, \cdots n$
For $t = 1, \cdots, T$
1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in [0,1]^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - \epsilon \cdot c_t(i))$

第一处修改扩展了代价的范围，第二处修改使得权重更保守，否则不存在完美策略时所有行动的权重全部归零

MWU 算法

Initialize weight $w_1(i) = 1, \forall i = 1, \cdots n$
For $t = 1, \cdots, T$
1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in [0,1]^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - \epsilon \cdot c_t(i))$

上图中给出的算法就是**乘性权重（Multiplicative Weights Update, MWU）**算法．算法的直观是，根据每个行动在之前阶段的表现来决定下一阶段的权重，即表现好的行动权重增加，表现差的行动权重减少．

MWU 算法

Initialize weight $w_1(i) = 1, \forall i = 1, \cdots n$
For $t = 1, \cdots, T$
1. Let $W_t = \sum_{i \in [n]} w_t(i)$, pick action i with probability $w_t(i)/W_t$
2. Observe cost vector $c_t \in [0,1]^n$
3. For any $i \in [n]$, update $w_{t+1}(i) = w_t(i) \cdot (1 - \epsilon \cdot c_t(i))$

上图中给出的算法就是**乘性权重（Multiplicative Weights Update, MWU）**算法．算法的直观是，根据每个行动在之前阶段的表现来决定下一阶段的权重，即表现好的行动权重增加，表现差的行动权重减少．

那么，乘性权重算法是否是无憾的呢？

定理

乘性权重算法在之前的问题设定下的遗憾至多为 $O(\sqrt{T \ln n})$．

MWU 算法分析

在证明这一定理之前，我们先简要说明这一上界是紧的，我们分别考虑 $\ln n$ 和 $\sqrt{T}$ 的存在必然性：

1. $\ln n$：考虑 $T \approx \ln(n-1)$ 构造如下代价序列：
   1. $t = 1$ 时，随机选择一半的行动有代价 1，其余行动有代价 0
   2. $t = 2, 3, \cdots, T$ 时，从剩余的完美行动中随机选择一半的行动有代价 1，其余行动有代价 0

   因为 $T < \ln n$，因此最终至少有一个行动是完美的，但是任意算法每一轮都要付出期望 $1/2$ 的代价，因此总代价为 $T/2$．注意这一代价甚至符合固定分布，恶意的对手则可能带来更坏的结果
2. $\sqrt{T}$：假设有两个行动，对手独立随机选择代价向量为 $(1,0)$ 或 $(0,1)$，故每一步的期望代价 $1/2$，累积 $T/2$，而离线下最优固定行动的期望累积代价仅为 $T/2 - b\sqrt{T}$，其中 b 是与 T 独立的常数，这是因为二项分布 $B(T, 1/2)$ 的标准差为 $\sqrt{T}/2$

MWU 算法分析（Cont'd）

定理

乘性权重算法在之前的问题设定下的遗憾至多为 $O(\sqrt{T \ln n})$.

接下来我们开始证明这一结论．首先我们需要一些观察，将期望代价与权重的下降关联起来：

- 第 t 轮的期望代价为：$\overline{C}_t = \sum_{i=1}^{n} p_t(i)c_t(i) = \dfrac{\sum_{i=1}^{n} w_t(i)c_t(i)}{W_t}$
- 事实上这正比于总权重的下降，即 $\sum_{i=1}^{n} \varepsilon w_t(i)c_t(i) = \varepsilon W_t \overline{C}_t$

因此我们的证明思想是：分析总权重下降的速度

MWU 算法分析（Cont'd）

引理 1

$W_{t+1} \leqslant W_t \cdot e^{-\varepsilon \overline{C}_t}$，其中 $W_t = \sum_{i=1}^{n} w_t(i)$ 是第 t 轮的总权重，$\overline{C}_t$ 是第 t 轮的期望代价.

MWU 算法分析（Cont'd）

引理 1

$W_{t+1} \leqslant W_t \cdot e^{-\varepsilon \overline{C}_t}$，其中 $W_t = \sum_{i=1}^{n} w_t(i)$ 是第 t 轮的总权重，$\overline{C}_t$ 是第 t 轮的期望代价.

证明：直接根据权重更新法则即可证明：

$$
\begin{aligned}
W_{t+1} &= \sum_{i=1}^{n} w_{t+1}(i) = \sum_{i=1}^{n} w_t(i) \cdot (1 - \varepsilon c_t(i)) \\
&= W_t - \varepsilon \sum_{i=1}^{n} w_t(i) c_t(i) = W_t - \varepsilon W_t \overline{C}_t = W_t(1 - \varepsilon \overline{C}_t) \\
&\leqslant W_t \cdot e^{-\varepsilon \overline{C}_t}
\end{aligned}
$$

MWU 算法分析（Cont'd）

引理 1

$W_{t+1} \leqslant W_t \cdot e^{-\varepsilon \overline{C}_t}$，其中 $W_t = \sum_{i=1}^{n} w_t(i)$ 是第 t 轮的总权重，$\overline{C}_t$ 是第 t 轮的期望代价.

推论 1

$$W_{T+1} \leqslant n e^{-\varepsilon \sum_{t=1}^{T} \overline{C}_t}.$$

证明：不断应用引理 1，我们有

$$\begin{aligned} W_{T+1} &\leqslant W_T \cdot e^{-\varepsilon \overline{C}_T} \leqslant W_{T-1} \cdot e^{-\varepsilon(\overline{C}_{T-1}+\overline{C}_T)} \leqslant \cdots \\ &\leqslant W_1 \cdot e^{-\varepsilon \sum_{t=1}^{T} \overline{C}_t} = n e^{-\varepsilon \sum_{t=1}^{T} \overline{C}_t} \end{aligned}$$

MWU 算法分析（Cont'd）

引理 2

对于任意的行动 i，都有 $W_{T+1} \geqslant e^{-T\varepsilon^2} \cdot e^{-\varepsilon \sum_{t=1}^{T} c_t(i)}$.

MWU 算法分析（Cont'd）

引理 2

对于任意的行动 i，都有 $W_{T+1} \geqslant e^{-T\varepsilon^2} \cdot e^{-\varepsilon \sum_{t=1}^{T} c_t(i)}$.

证明：这一步的放缩比较激进，但非常有效：

$$
\begin{aligned}
W_{T+1} \geqslant w_{T+1}(i) &= w_1(i) \cdot \prod_{t=1}^{T}(1-\varepsilon c_t(i)) \\
&\geqslant \prod_{t=1}^{T} e^{-\varepsilon c_t(i) - \varepsilon^2 c_t^2(i)} \\
&\geqslant e^{-T\varepsilon^2} \cdot e^{-\varepsilon \sum_{t=1}^{T} c_t(i)}
\end{aligned}
$$

其中第二行的不等号使用了 $1-\delta \geqslant e^{-\delta-\delta^2}$，第三行直接将 $c_t^2(i)$ 项放大成 1

MWU 算法分析（Cont'd）

推论 1 和引理 2

- $W_{T+1} \leqslant n e^{-\varepsilon \sum_{t=1}^{T} \overline{C}_t}$
- 对于任意的行动 i，都有 $W_{T+1} \geqslant e^{-T\varepsilon^2} \cdot e^{-\varepsilon \sum_{t=1}^{T} c_t(i)}$

综合推论 1 和引理 2，我们有 $e^{-T\varepsilon^2} \cdot e^{-\varepsilon \sum_{t=1}^{T} c_t(i)} \leqslant n e^{-\varepsilon \sum_{t=1}^{T} \overline{C}_t}$，这等价于

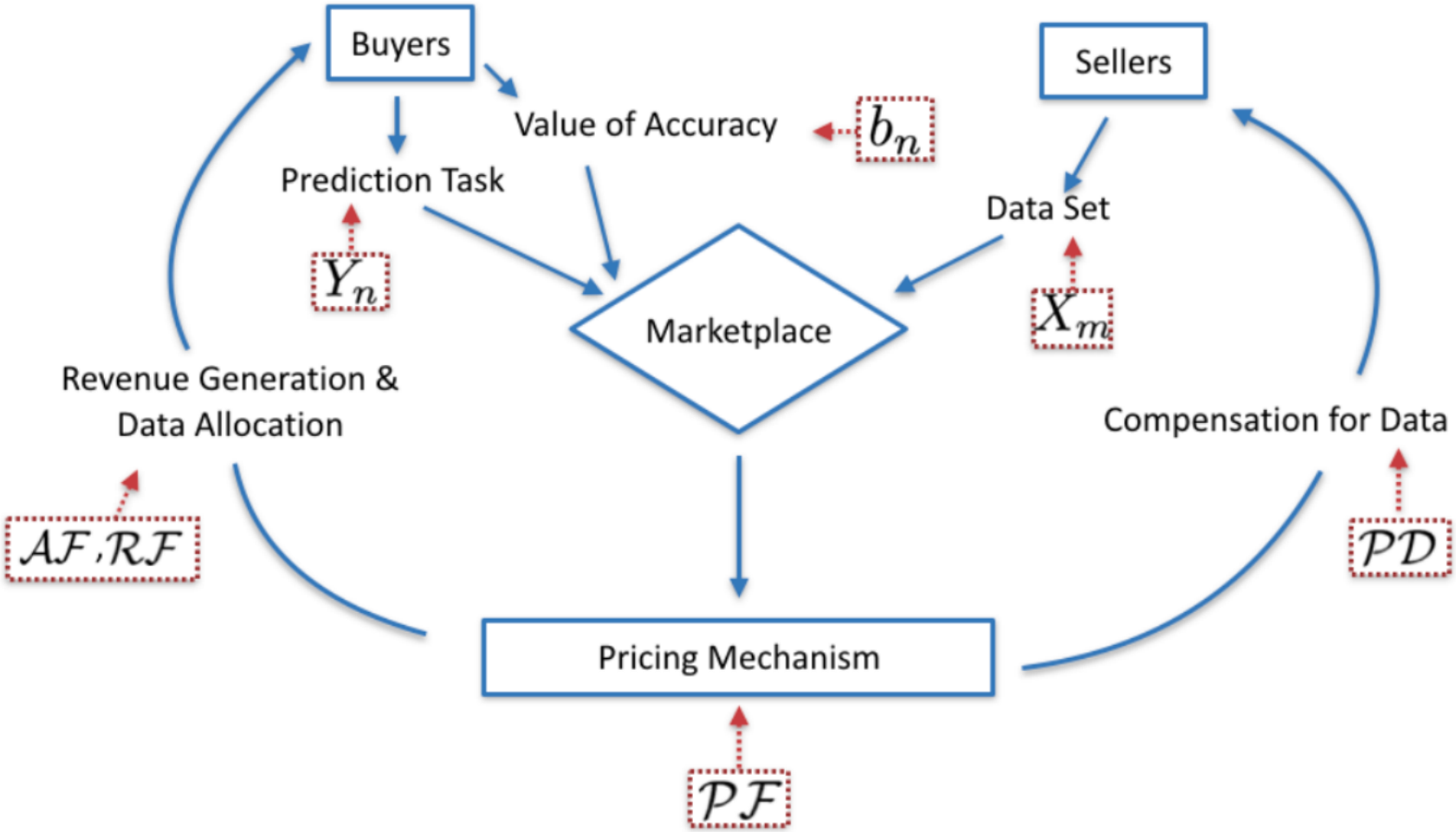
$$-T\varepsilon^2 - \varepsilon \sum_{t=1}^{T} c_t(i) \leqslant \ln n - \varepsilon \sum_{t=1}^{T} \overline{C}_t$$

重新整理有 $\sum_{t=1}^{T} \overline{C}_t - \sum_{t=1}^{T} c_t(i) \leqslant \frac{\ln n}{\varepsilon} + T\varepsilon$. 取 $\varepsilon = \sqrt{\frac{\ln n}{T}}$，我们有 $\sum_{t=1}^{T} \overline{C}_t - \min_i \sum_{t=1}^{T} c_t(i) \leqslant 2\sqrt{T \ln n}$，即遗憾至多为 $O(\sqrt{T \ln n})$.

MWU 的评注

- 如果我们使用 $w_{t+1}(i) = w_t(i) \cdot e^{-\varepsilon c_t(i)}$，因为 $e^{-\varepsilon} \approx 1 - \varepsilon$，因此也是可以进行类似的分析的；
- MWU 算法的应用非常广泛
  - 接下来我们会看到在定价中的应用
  - 无憾动力学与零和博弈均衡
  - 无憾动力学与粗糙相关均衡

MWU 在数据定价中的应用



[2] Agarwal, A., Dahleh, M.A., & Sarkar, T. (2018). A Marketplace for Data: An Algorithmic Solution. Proceedings of the 2019 ACM Conference on Economics and Computation.

MWU 在数据定价中的应用（Cont'd）

市场整体运行流程：

1. 数据市场设置此次交易的数据价格；
2. 买家到达数据市场，提出模型训练需求，同时提供投标
3. 数据卖家提供模型训练的数据，根据买家的投标和数据的市场价格向训练出的模型添加噪声
4. 市场决定向买家收取的费用，同时将费用公平分配给数据卖家

MWU 在数据定价中的应用（Cont'd）

市场整体运行流程：

1. 数据市场设置此次交易的数据价格；
2. 买家到达数据市场，提出模型训练需求，同时提供投标
3. 数据卖家提供模型训练的数据，根据买家的投标和数据的市场价格向训练出的模型添加噪声
4. 市场决定向买家收取的费用，同时将费用公平分配给数据卖家

解决方案：

1. 买家报价：训练中添加噪声符合迈尔森支付公式，因此买家的报价满足诚实性
2. 公平分配：Shapley value
3. 价格设置：最大化数据利润，即最小化遗憾，使用 MWU 算法

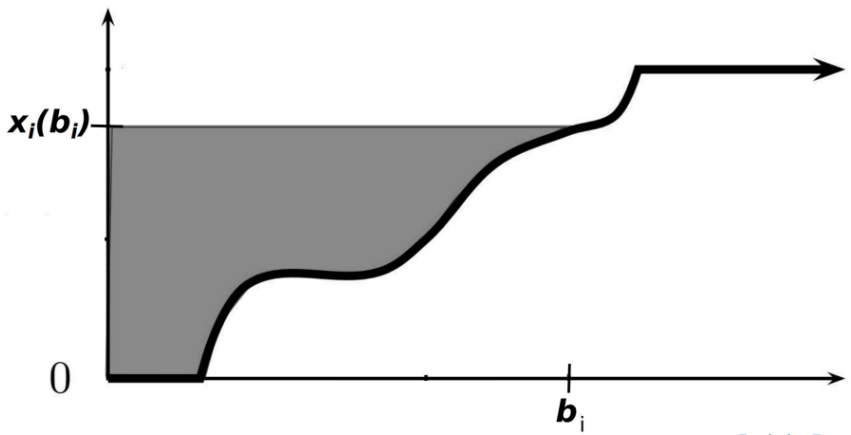
迈尔森引理

为了描述机器学习模型市场中诚实报价机制，我们需要首先证明迈尔森引理，因为这一机制完全是迈尔森引理的推论．回忆上一次中的单参数环境：收集所有智能体的出价 $\boldsymbol{b} = (b_1, \cdots, b_n)$，物品分配规则 $\boldsymbol{x}(\boldsymbol{b})$，支付规则 $\boldsymbol{p}(\boldsymbol{b})$，则迈尔森引理如下：

定理 (迈尔森引理)

在单参数环境下，上述机制是 DSIC 的当且仅当

- x 是单调的，即 $x_i(\boldsymbol{b})$ 是 b_i 的单调不减函数（报价越高，获得物品的概率越高）；
- $p_i(\boldsymbol{b}) = b_i \cdot x_i(\boldsymbol{b}) - \int_0^{b_i} x_i(\boldsymbol{b}) \mathrm{d}b_i$



迈尔森引理的证明

定理 (迈尔森引理)

在单参数环境下，上述机制是 DSIC 的当且仅当

- x 是单调的，即 $x_i(\boldsymbol{b})$ 是 b_i 的单调不减函数（报价越高，获得物品的概率越高）；
- $p_i(\boldsymbol{b}) = b_i \cdot x_i(\boldsymbol{b}) - \int_0^{b_i} x_i(\boldsymbol{b})\mathrm{d}b_i$

定理“当”的部分直接验证即可，没有困难；难点在于如何导出这一具体的机制：我们需要用到一个很聪明的交换技巧．假设机制 $(\boldsymbol{x}, \boldsymbol{p})$ 是 DSIC 的，有两个任意的变量 $0 \leqslant y < z$．一种可能的情况是，智能体 i 的估值是 z，提交报价是 y，此时 DSIC 要求

$$z \cdot x(z) - p(z) \geqslant z \cdot x(y) - p(y)$$

另一种情况是，智能体 i 的估值是 y，提交报价是 z，此时要求

$$y \cdot x(y) - p(y) \geqslant y \cdot x(z) - p(z)$$

迈尔森引理的证明（Cont'd）

上述两式移项、组合后可以得到

$$z \cdot (x(y) - x(z)) \leqslant p(y) - p(z) \leqslant y \cdot (x(y) - x(z))$$

实际上由这一不等式的左右两端已经可以看出，支付规则必须是单调的．然后我们可以对三项分别除以 $y - z$，然后另 $y \to z$ 可得

$$y \cdot x'(y) = p'(y)$$

两边对 y 积分可得

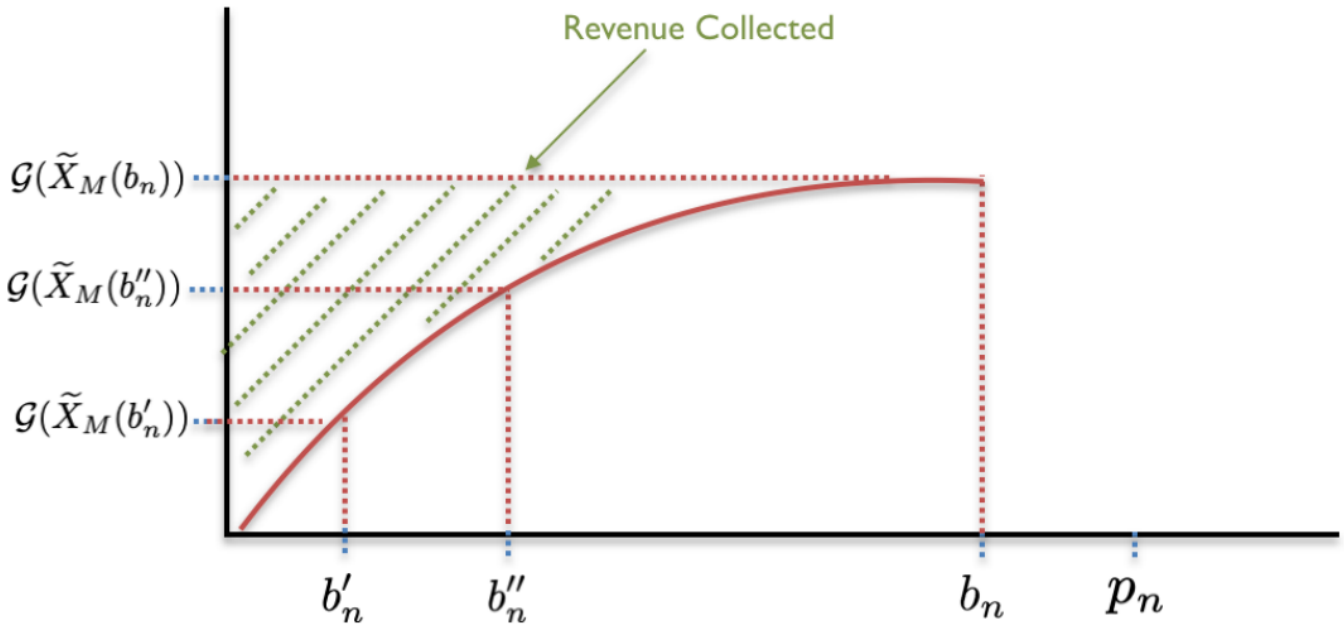
$$\int_0^{b_i} p'(y)\mathrm{d}y = \int_0^{b_i} yx'(y)\mathrm{d}y$$

展开即可得到迈尔森引理．

迈尔森引理与机器学习模型定价

我们能从迈尔森引理中得到什么有关模型定价的启示吗?

- 模型的分配 $\boldsymbol{x}$ 如何随着报价的升高而增加? 如果我们将分配值视为模型学习准确率，那么报价越高，我们往模型中添加的噪声越小，通过这样的方式可以实现单调的分配函数
- 然后支付函数仍然是 $p_i(\boldsymbol{b}) = b_i \cdot x_i(\boldsymbol{b}) - \int_0^{b_i} x_i(\boldsymbol{b})\mathrm{d}b_i$，这样可以实现诚实报价占优



在线学习与重复博弈

- 重复博弈：同一个博弈重复进行很多轮，例如多局的石头剪刀布
- 在现实中，我们的确像在线学习一样进行重复博弈
  - 我们尝试分析对手以往的行为，以便预测对手的下一步行为，当然可能对手行为是有概率分布的

重复零和博弈与无憾参与者

我们首先研究重复零和博弈，每个参与者都希望无憾，基本设定如下：

1. 假设零和博弈的收益矩阵是 $U \in \mathbb{R}^{m \times n}$
2. 行参与者最大化收益矩阵的值，列参与者最小化收益矩阵的值
3. 博弈重复 T 轮，每轮参与者使用一个在线学习算法选择混合策略

重复零和博弈与无憾参与者

我们首先研究重复零和博弈，每个参与者都希望无憾，基本设定如下：

1. 假设零和博弈的收益矩阵是 $U \in \mathbb{R}^{m \times n}$
2. 行参与者最大化收益矩阵的值，列参与者最小化收益矩阵的值
3. 博弈重复 T 轮，每轮参与者使用一个在线学习算法选择混合策略

从行参与者的视角看，以下事件依次发生：

1. 选择一个策略集 $[m]$ 上的混合策略 x_t
2. 对手选择一个策略集 $[n]$ 上的混合策略 y_t
3. 选择行动 $i_t \sim x_t$，收益 $U(i_t, y_t) = \sum_{i=1}^{n} y_t(j) U(i_t, j)$
4. 行参与者学习到 y_t 以调整未来的策略

重复零和博弈与无憾参与者

我们首先研究重复零和博弈，每个参与者都希望无憾，基本设定如下：

1. 假设零和博弈的收益矩阵是 $U \in \mathbb{R}^{m \times n}$
2. 行参与者最大化收益矩阵的值，列参与者最小化收益矩阵的值
3. 博弈重复 T 轮，每轮参与者使用一个在线学习算法选择混合策略

从行参与者的视角看，以下事件依次发生：

1. 选择一个策略集 $[m]$ 上的混合策略 x_t
2. 对手选择一个策略集 $[n]$ 上的混合策略 y_t
3. 选择行动 $i_t \sim x_t$，收益 $U(i_t, y_t) = \sum_{i=1}^{n} y_t(j) U(i_t, j)$
4. 行参与者学习到 y_t 以调整未来的策略

列参与人：对称地，只是将 $U(i,j)$ 作为其代价而非收益

重复零和博弈与无憾参与者（Cont'd）

与在线学习的区别：收益或代价并不是任意给定的，而是根据对手的选择得到的

- 行参与人的期望收益（列参与人的期望代价）为 $\sum_{i=1}^{T} U(x_t, y_t)$
  - 回忆：$U(x_t, y_t) = \sum_{i=1}^{m} \sum_{j=1}^{n} x_t(i) y_t(j) U(i,j) = (x_t)^{\mathrm{T}} U y_t$
- 行参与者的遗憾：$\max_{i \in [m]} \sum_{t=1}^{T} U(i, y_t) - \sum_{t=1}^{T} U(x_t, y_t)$
- 列参与者的遗憾：$\sum_{t=1}^{T} U(x_t, y_t) - \min_{j \in [n]} \sum_{t=1}^{T} U(x_t, j)$

从无憾算法到最小最大定理

接下来我们利用无憾算法存在性（MWU）来证明最小最大定理：假设行列参与人都使用无憾在线算法，对于行参与者，我们有

$$R_T^{row} = \max_{i\in[m]} \sum_{t=1}^{T} U(i, y_t) - \sum_{t=1}^{T} U(x_t, y_t)$$

这等价于

$$\frac{1}{T}\sum_{t=1}^{T} U(x_t, y_t) + \frac{R_T^{row}}{T} = \frac{1}{T}\max_{i\in[m]} \sum_{t=1}^{T} U(i, y_t)$$

从无憾算法到最小最大定理

接下来我们利用无憾算法存在性（MWU）来证明最小最大定理：假设行列参与人都使用无憾在线算法，对于行参与者，我们有

$$R_T^{row} = \max_{i\in[m]} \sum_{t=1}^{T} U(i, y_t) - \sum_{t=1}^{T} U(x_t, y_t)$$

这等价于

$$\frac{1}{T}\sum_{t=1}^{T} U(x_t, y_t) + \frac{R_T^{row}}{T} = \frac{1}{T}\max_{i\in[m]} \sum_{t=1}^{T} U(i, y_t)$$

而我们知道

$$\frac{1}{T}\max_{i\in[m]} \sum_{t=1}^{T} U(i, y_t) = \max_{i\in[m]} U\left(i, \frac{1}{T}\sum_{t=1}^{T} y_t\right) \geqslant \min_{y\in\Delta([n])} \max_{i\in[m]} U(i, y)$$

从无憾算法到最小最大定理（Cont'd）

因此，对于行参与者，我们有

$$\frac{1}{T}\sum_{t=1}^{T} U(x_t, y_t) + \frac{R_T^{row}}{T} \geqslant \min_{y\in\Delta([n])} \max_{i\in[m]} U(i,y)$$

同理，对于列参与者，我们可以得到

$$\frac{1}{T}\sum_{t=1}^{T} U(x_t, y_t) - \frac{R_T^{col}}{T} \leqslant \max_{x\in\Delta([m])} \min_{j\in[n]} U(x,j)$$

令 $T \to \infty$，无遗憾算法保证 $\frac{R_T^{row}}{T} \to 0$ 和 $\frac{R_T^{col}}{T} \to 0$，因此有

$$\min_{y\in\Delta([n])} \max_{i\in[m]} U(i,y) \leqslant \max_{x\in\Delta([m])} \min_{j\in[n]} U(x,j)$$

另一半不等号此前我们已经证明，综合即可得到最小最大定理

收敛于纳什均衡

定理

假设行列参与人都使用无憾算法，使用策略序列 $\{x_t\}_{t=1}^T$ 和 $\{y_t\}_{t=1}^T$，则 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，$\left(\frac{1}{T}\sum_{t=1}^T x_t, \frac{1}{T}\sum_{t=1}^T y_t\right)$ 收敛于纳什均衡.

收敛于纳什均衡

定理

假设行列参与人都使用无憾算法，使用策略序列 $\{x_t\}_{t=1}^T$ 和 $\{y_t\}_{t=1}^T$，则 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，$\left(\frac{1}{T}\sum_{t=1}^T x_t, \frac{1}{T}\sum_{t=1}^T y_t\right)$ 收敛于纳什均衡.

证明：根据

$$\frac{1}{T}\sum_{t=1}^T U(x_t, y_t) + \frac{R_T^{row}}{T} = \max_{i\in[m]} U\left(i, \frac{1}{T}\sum_{t=1}^T y_t\right) \geqslant \min_{y\in\Delta([n])} \max_{i\in[m]} U(i, y)$$

收敛于纳什均衡

定理

假设行列参与人都使用无憾算法，使用策略序列 $\{x_t\}_{t=1}^T$ 和 $\{y_t\}_{t=1}^T$，则 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，$\left(\frac{1}{T}\sum_{t=1}^T x_t, \frac{1}{T}\sum_{t=1}^T y_t\right)$ 收敛于纳什均衡.

证明：根据

$$\frac{1}{T}\sum_{t=1}^T U(x_t, y_t) + \frac{R_T^{row}}{T} = \max_{i\in[m]} U\left(i, \frac{1}{T}\sum_{t=1}^T y_t\right) \geqslant \min_{y\in\Delta([n])} \max_{i\in[m]} U(i, y)$$

当 $T \to \infty$ 时，不等号变成等于号，因此 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，并且 $\frac{1}{T}\sum_{t=1}^T y_t$ 就是最小最大值对应的 y，同理 $\frac{1}{T}\sum_{t=1}^T x_t$ 就是最大最小值对应的 x，因此收敛于纳什均衡（回忆零和博弈纳什均衡的等价性）.

收敛于纳什均衡

定理

假设行列参与人都使用无憾算法，使用策略序列 $\{x_t\}_{t=1}^T$ 和 $\{y_t\}_{t=1}^T$，则 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，$\left(\frac{1}{T}\sum_{t=1}^T x_t, \frac{1}{T}\sum_{t=1}^T y_t\right)$ 收敛于纳什均衡.

证明：根据

$$\frac{1}{T}\sum_{t=1}^T U(x_t, y_t) + \frac{R_T^{row}}{T} = \max_{i\in[m]} U\left(i, \frac{1}{T}\sum_{t=1}^T y_t\right) \geqslant \min_{y\in\Delta([n])} \max_{i\in[m]} U(i, y)$$

当 $T \to \infty$ 时，不等号变成等于号，因此 $\frac{1}{T}\sum_{t=1}^T U(x_t, y_t)$ 收敛于博弈的值，并且 $\frac{1}{T}\sum_{t=1}^T y_t$ 就是最小最大值对应的 y，同理 $\frac{1}{T}\sum_{t=1}^T x_t$ 就是最大最小值对应的 x，因此收敛于纳什均衡（回忆零和博弈纳什均衡的等价性）.

应用：德州扑克战胜人类的算法设计，但要注意提升算法的收敛速率

MWU 与粗糙相关均衡

粗糙相关均衡（coarse correlated equilibrium）是相关均衡的一个扩展，此时参与人只知道自己被推荐策略的概率，而不知道自己被推荐的具体策略，并且不偏离推荐：

$$\sum_{s\in S} u_i(s)p_i(s) \geqslant \sum_{s\in S} u_i(s', s_{-i})p_i(s), \forall s_i' \in S_i$$

MWU 与粗糙相关均衡

粗糙相关均衡（coarse correlated equilibrium）是相关均衡的一个扩展，此时参与人只知道自己被推荐策略的概率，而不知道自己被推荐的具体策略，并且不偏离推荐：

$$\sum_{s\in S} u_i(s)p_i(s) \geqslant \sum_{s\in S} u_i(s', s_{-i})p_i(s), \forall s'_i \in S_i$$

同样地，博弈重复 T 轮，每轮参与者使用一个在线学习算法选择混合策略．从参与人 i 的角度来看，以下事件在第 t 轮依次发生：

1. 选择策略集 S_i 上的混合策略 $\sigma_i^t \in \Delta(S_i)$
2. 其它参与人 $j \neq i$ 选择混合策略 $\sigma_j^t \in \Delta(S_j)$
3. 参与人 i 得到期望效用 $u_i(\sigma_i^t, \sigma_{-i}^t) = \mathbf{E}_{s\sim\sigma^t}[u_i(s)]$
4. 参与人 i 学习到 σ_{-i}^t 以调整未来的策略

MWU 与粗糙相关均衡（Cont'd）

在这一设定下，每个参与人 i 的遗憾值为

$$R_T^i = \max_{s_i \in S_i} \sum_{t=1}^{T} u_i(s_i, \sigma_{-i}^t) - \sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}^t)$$

MWU 与粗糙相关均衡（Cont'd）

在这一设定下，每个参与人 i 的遗憾值为

$$R_T^i = \max_{s_i \in S_i} \sum_{t=1}^{T} u_i(s_i, \sigma_{-i}^t) - \sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}^t)$$

定理

假设每个参与人 i 使用无憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于粗糙相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

MWU 与粗糙相关均衡（Cont'd）

在这一设定下，每个参与人 i 的遗憾值为

$$R_T^i = \max_{s_i \in S_i} \sum_{t=1}^{T} u_i(s_i, \sigma_{-i}^t) - \sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}^t)$$

定理

假设每个参与人 i 使用无憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于粗糙相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

- 事实上 $\pi^T(s)$ 就是纯策略策略向量 s 在 T 轮中被选择的平均概率
- 这一定理也给出了计算粗糙相关均衡的非常简单的方法

MWU 与粗糙相关均衡（Cont'd）

定理

假设每个参与人 i 使用无憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于粗糙相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

证明：在 π^T 下，参与人 i 的期望效用为

$$\begin{aligned}\sum_{s\in S}\pi^T(s)u_i(s) &= \sum_{s\in S}\left[\frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i)\right]u_i(s)\\ &= \frac{1}{T}\sum_{t=1}^{T}\sum_{s\in S}\prod_{i=1}^{n}\sigma_i^t(s_i)u_i(s)\\ &= \frac{1}{T}\sum_{t=1}^{T}u_i(\sigma_i^t,\sigma_{-i}^t)\end{aligned}$$

MWU 与粗糙相关均衡（Cont'd）

定理

假设每个参与人 i 使用无憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于粗糙相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

证明（续）：根据无憾算法可知，对于任意的参与人 i，有

$$\lim_{T\to\infty}\frac{R_T^i}{T} = \lim_{T\to\infty}\left(\frac{1}{T}\max_{s_i\in S_i}\sum_{t=1}^{T}u_i(s_i,\sigma_{-i}^t) - \frac{1}{T}\sum_{t=1}^{T}u_i(\sigma_i^t,\sigma_{-i}^t)\right) = 0$$

因此我们可以推出

$$\frac{1}{T}\sum_{t=1}^{T}u_i(\sigma_i^t,\sigma_{-i}^t) \geqslant \frac{1}{T}\sum_{t=1}^{T}u_i(s_i,\sigma_{-i}^t), \forall s_i \in S_i$$

这就表明 π^T 的确收敛于粗糙相关均衡

从外部遗憾到交换遗憾

我们可以进一步考察（外部）遗憾：

$$
\begin{aligned}
R_T &= \mathbf{E}_{i_t \sim p_t}\left[\sum_{t=1}^{T} c_t(i_t)\right] - \min_{j\in[n]} \sum_{t=1}^{T} c_t(j) \\
&= \sum_{t=1}^{T}\sum_{i=1}^{n} c_t(i)p_t(i) - \min_{j\in[n]} \sum_{t=1}^{T} c_t(j) \\
&= \max_{j\in[n]} \left[\sum_{t=1}^{T}\sum_{i=1}^{n} c_t(i)p_t(i) - \sum_{t=1}^{T} c_t(j)\right] \\
&= \max_{j\in[n]} \sum_{t=1}^{T}\sum_{i=1}^{n} [c_t(i) - c_t(j)]p_t(i)
\end{aligned}
$$

从外部遗憾到交换遗憾

我们可以进一步考察（外部）遗憾：

$$
\begin{aligned}
R_T &= \mathbf{E}_{i_t \sim p_t}\left[\sum_{t=1}^{T} c_t(i_t)\right] - \min_{j\in[n]} \sum_{t=1}^{T} c_t(j) \\
&= \sum_{t=1}^{T}\sum_{i=1}^{n} c_t(i)p_t(i) - \min_{j\in[n]} \sum_{t=1}^{T} c_t(j) \\
&= \max_{j\in[n]}\left[\sum_{t=1}^{T}\sum_{i=1}^{n} c_t(i)p_t(i) - \sum_{t=1}^{T} c_t(j)\right] \\
&= \max_{j\in[n]} \sum_{t=1}^{T}\sum_{i=1}^{n} [c_t(i) - c_t(j)]p_t(i)
\end{aligned}
$$

最后的括号表明了外部遗憾的本质：我们可以把所有的行动都交换到一个固定的行动 j，从而获得更低的遗憾

从外部遗憾到交换遗憾（Cont'd）

我们将多对一的交换拓展到多对多的交换，就可以得到**交换遗憾（swap regret）**的概念：

定义

对于一个多轮的博弈，我们定义交换遗憾为

$$swR_T = \max_{\pi \in \Pi} \sum_{t=1}^{T} \sum_{i=1}^{n} [c_t(i) - c_t(\pi(i))] p_t(i)$$

其中 Π 是所有的 $[n]$ 到 $[n]$ 的映射（不一定是双射）

1. 外部遗憾是交换遗憾的特例，即取了所有映射值唯一的情况
2. 一共有 n^n 种交换映射
3. 注意交换后也不一定是全局最优的行动序列，因为每一步的交换函数都是固定的

交换遗憾的性质

定理

1. 对于任意的在线学习算法，有 $R_T \leqslant swR_T$
2. 对于任意的策略序列 $p_1, \cdots, p_T$，最优的交换函数 π^* 满足对任意的 $i \in [n]$，有

$$\pi^*(i) = \arg\max_{j \in [n]} \sum_{t=1}^{T} [c_t(i) - c_t(j)] p_t(i)$$

交换遗憾的性质

定理

1. 对于任意的在线学习算法，有 $R_T \leqslant swR_T$
2. 对于任意的策略序列 $p_1, \cdots, p_T$，最优的交换函数 π^* 满足对任意的 $i \in [n]$，有

$$\pi^*(i) = \arg\max_{j \in [n]} \sum_{t=1}^{T} [c_t(i) - c_t(j)] p_t(i)$$

1. 外部遗憾是交换遗憾的特例，因此结论显然成立
2. 因为每个 i 都是独立的，因此各自做最优选择即可

交换遗憾与相关均衡

定理

假设每个参与人 i 使用无交换遗憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

- 事实上这个推荐序列和粗糙相关均衡的推荐序列是一样的，因此关键在于无交换遗憾是一个更强的性质，因此能收敛到一个更强的均衡，具体证明留作习题
- 回忆之前的一道作业题

交换遗憾与相关均衡

定理

假设每个参与人 i 使用无交换遗憾算法得到决策序列 $\{\sigma_i^t\}_{t=1}^T$，则下面的推荐策略 π^T 收敛于相关均衡：$\pi^T(s) = \frac{1}{T}\sum_{t=1}^{T}\prod_{i=1}^{n}\sigma_i^t(s_i), \forall s \in S.$

- 事实上这个推荐序列和粗糙相关均衡的推荐序列是一样的，因此关键在于无交换遗憾是一个更强的性质，因此能收敛到一个更强的均衡，具体证明留作习题
- 回忆之前的一道作业题
- 但无交换遗憾算法是否存在呢?

定理

一个具有外部遗憾 R 的在线学习算法可以被转化为一个具有交换遗憾 nR 的在线学习算法，其中 n 是参与人策略数目.

感兴趣的同学可以参考《算法博弈论二十讲》第 18 讲.