Dueling Bandits

传统的赌博机（Multi-armed Bandits）往往假设奖励仅与动作相关或与场景和动作的联合特征相关，并且可以量化，然而存在这一要求无法满足的实际情况。**同样对于一个动作集合，假设其绝对奖励是不可知的，但同时选择多个动作，可以得到它们之间的比较结果**，这一类问题需要完全不同的算法
一个例子是商品的受欢迎程度比较。假设两个售货机分别出售百事和可口两种可乐，抽取它们多天的销售数据可以得到两种可乐的每日销量。如果采用绝对数量来衡量两个商标的受欢迎程度，很容易受到特殊情况影响，例如某一天可口的消费者非常渴而购买了大量饮料，使得可口的销售总量大幅上升。长期对比二者的相对销量则会更合理
下面建模并提出算法

Model of Dueling Bandits

假设一定有一个作为Condorcet Winner的Arm（可以在两两对决中胜过其它所有竞争者的Arm），并且Arms可以按照好坏顺序排列，不会出现$A > B, B > C, C > A$的情况
与MBA一样，目的是最小化$R(T) = \Sigma_{t=1}^{T} r(t)$，其中$r = r_{arm-i} + r_{arm-j}$，其数值计算在不同的算法中可以不同
为了量化每个Arm的好坏，通常使用对称矩阵Preference Matrix记录每个Arm战胜其它Arm的概率，例如

	A	B	C	D	E	F
A	0	0.03	0.04	0.06	0.10	0.11
B	-0.03	0	0.03	0.05	0.08	0.11
C	-0.04	-0.03	0	...	...	...
D	-0.06	-0.05	...	0	...	...
E	-0.10	-0.08	...	...	0	...
F	-0.11	-0.11	...	...	...	0

其中A是最好的Arm，A打败B的概率是0.53，A打败E的概率是0.6，矩阵中的每个元素Δ表示其所在的行的Arm击败其所在的列的Arm的概率与0.5的差，$P = \Delta + 0.5$，Arm的排列具有单调性，许多算法都使用Preference Matrix计算Regret

Algorithms of Dueling Bandits

基本上有两类算法，一种是对称选择，每次等价选择两个Arm进行比较，另一种是非对称的，一般随机选择一个Arm与目前最优的Arm进行比较
另外，Dueling Bandits也存在Contextual/Adversarial的情况，暂时不考虑

Beat the Mean

选择上述Preference Matrix计算Regret，定义Borda score为$S_i = \Sigma_j P_{ij}$，其中S_i是Arm_i的Borda score，则Condorcet Winner具有最高的Borda score，这保证了BtM算法的收敛性

```
初始化Bandits的K个Arms：B={b_1,b_2,...,B_K}
设置超参数迭代轮数T，最小比较次数上限N，置信半径C
```

```
维护三个K维向量N，W，P，其中N为每个b（Arm）的比较次数，W为每个b的胜利次数，P=W/N为每个b的胜率
设置最小的比较次数n=min(N)
当t<T且n<N时迭代{
取比较次数最少的b
随机选择b'
比较b与b'
根据结果更新N，W，P
t+=1
取此时最高和最低的胜率，若其差比C更大{
取出胜率最低的b'
删除其它所有b与b'的比较记录
从B中删除b'
}
}
```

BtM是典型的非对称算法，具有时间上界$O(\frac{\gamma^7 KlogT}{min(\Delta_{1j})})$，其中gamma是衡量Arm差距的参数，K是Arm数量，T是迭代次数上限

Sparring/Self-Sparring

这两个算法是典型的对称算法
Sparring使用两个典型算法（如Thompson Sampling），每次各自选择一个Arm比较，然后将胜负反馈给两个算法，而Self-Sparring只使用一个典型算法选择，Self-Sparring的上界是$O(\frac{KlogT}{\epsilon})$，其中$\epsilon$是最好和次好的Arm的差距
对称的算法应用场景更广，可以将两个算法视为两个对手，选择Arm进行对决可以视为Zero-sum Game，二者均在对决中得到优化

Reference

[Dueling-Bandits综述]

[Slides]