LiuUCB 算法：

1. Multi-armed bandit Formulation

我们先形式化的定义 K-armed contextual bandit 问题，将个性化新闻推荐建模为具有 context 信息的多臂赌博机问题：本算法不断的对离散的 trials 进行处理。在 trial t，算法可以观察到当前的用户 u_t，所有的 arms（actions）$a \in A_t$，和包含该用户信息和所有的 arms 信息的向量 $X_{t,a}$（context）；基于前面 trial 的 payoff，本算法选择一个 arm a_t，得到 payoff $r_{t,at}$，该值的期望取决于当前用户和选择的 arm；最后利用新观察到的信息来提升选择 arm 的策略。在所有的 T 个 trial 中，total T-trial payoff 定义为 $\sum_{t=1}^{T} r_{t,a_t}$，optimal expected T-trial payoff 定义为 $\mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right]$，其中 a^*_t 是在 t trial 时能获得最大 payoff 的臂。本算法的目标是最大化 total payoff 的期望，也相当于最小化 regret：

$$R_A(T) \stackrel{\text{def}}{=} \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right]$$

在推荐场景中，我们将待推荐的文章视为 arms，如果推荐的文章被选择，则得到 payoff 1 否则得到 0（每次只能利用被选择文章的 payoff）。这样一个文章的 payoff 的期望正好是这篇文章的点击率（事实上，payoff 就是一个 action 的 reward，用户作为环境，用户是否点击就是环境给的不同 reward 的概率，reward 的期望为 P(点击)*1+P(不点击)*0= P(点击)），所以选择文章的使其点击率最大化相当于最大化用户点击推荐文章的次数，也相当于最大 total expect payoff。

对于 context-free K-armed bandit 问题，已有的方法比如ϵ-greedy 方法，每次 estimate 每个 action 的 average payoff：$\mu_{t,a}$，然后基于此以ϵ-greedy 的策略采样，由于每个 action 都会被采样无数次，$\mu_{t,a}$ 最后会收敛到 action 的真实 value（reward 的期望值）μ_t，每一步的 regret 会收敛到 0。这种探索是无指导的，另一种方法称为 upper confidence bound algorithm，更好的平衡了 E&E 问题，每一步 t，我们不仅得到 average payoff：$\mu_{t,a}$，还得到对应的置信间隔 $c_{t,a}$，$|\hat{\mu}_{t,a} - \mu_a| < c_{t,a}$ 具有高概率成立，然后选择 $\arg\max_a (\hat{\mu}_{t,a} + c_{t,a})$，则对于置信间隔高的 arm 会进行更多的探索，来缩小它的置信区间。

总体上，我们的 task 是预计每个 arm 的 value，即 estimate 每个 arm reward 的期望 $E[r_a]$，然后根据这个 estimate 选择值最大的 arm（同时加入一些 exploration，比如置信间隔或者ϵ-greedy，或者一些 off-policy 的方法）。对于 context-free 的推荐场景（相当于只有一个用户），我们在每一个时间步 t 对每个 arm 的 value 有一个新的估计 $E[r_{t,a}]$（t 步推荐的 arm 更新，其他 arm 的 value 为 $E[r_{t-1,a}]$），这个估计为用户 click 的次数\推荐给用户的次数。目前的场景，由于我们有了 context，我们用一个模型得到 $E[r_{t,a}]$，本文中具体建模为与 context $X_{t,a}$ 有关的线性模型。

2. LinUCB with Disjoint Linear Models

如果 payoff 模型是线性的，置信间隔可以以封闭的形式有效计算，这种算法称为 LinUCB，下面先介绍不相交的线性模型，再介绍混合线性模型。

假设某个 arm 的预期 payoff 是与 d 维特征 $X_{t,a}$ 成线性关系的，基于系数向量 θa*: $\mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_a^*$。本模型是不相交的（disjoint），因为不同的 arm 没有共享参数。在时间 t，对于 arm a，我们已经有了训练数据(D_a,c_a)，其中 D_a 是一个 m*d 的矩阵，在前 t 个时间步中 m 次推荐 a，m 个 contexts 组成矩阵。c_a 是一个 m 维向量，包含每次用户是否 click 的信息。此时要估计 $r_{t,a}$ 的 reward，我们要先得到 θa*的估计值，对其进行贝叶斯点估计（也就是在最小二乘的基础上加一个 l2 正则化项 https://blog.csdn.net/daunxx/article/details/51578787）。得到：

$$\hat{\boldsymbol{\theta}}_a = (\mathbf{D}_a^{\top}\mathbf{D}_a + \mathbf{I}_d)^{-1}\mathbf{D}_a^{\top}\mathbf{c}_a$$

得到 θ 后利用此刻的 context $x_{t,a}$ 我们可以估计 value。我们仍使用 UCB 方法，所以我们还需要计算一个置信间隔，已知下式被证明以很高的概率成立：

$$\left|\mathbf{x}_{t,a}^{\top}\hat{\boldsymbol{\theta}}_a - \mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}]\right| \le \alpha\sqrt{\mathbf{x}_{t,a}^{\top}(\mathbf{D}_a^{\top}\mathbf{D}_a + \mathbf{I}_d)^{-1}\mathbf{x}_{t,a}}$$

这样我们可以估计每一个 arm 的 value+置信间隔，我们选择

$$a_t \overset{\text{def}}{=} \arg\max_{a\in\mathcal{A}_t}\left(\mathbf{x}_{t,a}^{\top}\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{\mathbf{x}_{t,a}^{\top}\mathbf{A}_a^{-1}\mathbf{x}_{t,a}}\right)$$

其中$\mathbf{A}_a \overset{\text{def}}{=} \mathbf{D}_a^{\top}\mathbf{D}_a + \mathbf{I}_d$。

算法如下，注意对 θ 的估计是增量的更新的，见最后两行：

Algorithm 1 LinUCB with disjoint linear models.

0: Inputs: $\alpha \in \mathbb{R}_+$
1: **for** $t = 1, 2, 3, \ldots, T$ **do**
2: Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
3: **for all** $a \in \mathcal{A}_t$ **do**
4: **if** a is new **then**
5: $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
6: $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
7: **end if**
8: $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
9: $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^{\top}\mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^{\top}\mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
10: **end for**
11: Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
12: $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^{\top}$
13: $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
14: **end for**

3. evaluation methodology

要对不同的 bandit 方法（这里称为 policy）进行评估是有困难的，因为我们不能直接用它和环境 online 的交互，我们只拥有用其他 policy 得到的 offline 的数据。我们将其他 policy 和世界交互的数据分为很多个 event，每个 event 包含上下文信息，选择的 arm a，和得到的 payoff r_a，然后我们用 π 利用历史信息和当前 event 的上下文信息选择 arm，如果刚好选择 arm，则将这个 event 加入 π 的历史数据，否则跳过，最后用平均 reward 作为 π 的评估。

Algorithm 3 Policy_Evaluator.

Inputs: $T > 0$; policy π; stream of events
$h_0 \leftarrow \emptyset$ {An initially empty history}
$R_0 \leftarrow 0$ {An initially zero total payoff}
for $t = 1, 2, 3, \ldots, T$ **do**
  repeat
    Get next event $(\mathbf{x}_1, ..., \mathbf{x}_K, a, r_a)$
  until $\pi(h_{t-1}, (\mathbf{x}_1, ..., \mathbf{x}_K)) = a$
  $h_t \leftarrow \text{CONCATENATE}(h_{t-1}, (\mathbf{x}_1, ..., \mathbf{x}_K, a, r_a))$
  $R_t \leftarrow R_{t-1} + r_a$
end for
Output: R_T/T

引申，对于 POI 问题，由于在每个时间步具有 state，所以不能跳过真实数据的每一次 payoff（state 的转移），可以用具有随机性的 π 多次选择，选择到的 action 不是真实的下一个 action，就当做用户的 reward 为 0，选择到的是真的 action，则获得 reward（比如像本文被点击返回 1），并 state 进行转移。本文不能这样做是因为新闻点击率问题中，系统会把预测的新闻提供给用户，用户不点击意味着不感兴趣，我们无法将没有提供给用户的新闻当做用户不感兴趣的。而 POI 场景中，系统不主动提供地点，用户去的就是感兴趣的，没去的给 0 作为 reward（没去的都在用户的选择范围之内）。