強化学習勉強会

Lihong Li et al., 2010, A Contextual-Bandit Approach to Personalized News Article Recommendation

田中一樹

この論文を選んだ理由

- ちょっと前にスマホプッシュ配信にBanditを使った記事が出て少し盛り上がった記憶
    - 知っておきたい（？）
- Webサイトや広告メールにのせる記事や商品を実際に出しながらユーザごとに適した対応がしたい場合多そう
    - CTR(Click Through Rate)やConversion率を高めたい
    - Bandit使えるってなってるけど実際どういう風に特徴量とかアルゴリズム適用してるの？？

今日の目標

- Contextual-Banditアルゴリズムの雰囲気を知る
- 具体的なデータへの適用方法と有用性を知る
- どんなフィーチャやデータセット、状況に応用可能かふんわり考える

目次

WEBにおけるコンテンツ配信の難しさ(Motivation)

- **コンテンツが大量**にあり、常に追加、削除を繰り返しながら**変化**
  - 古いニュース記事の影響を徐々に消していきながら、速報の人気度を迅速に特定するとか

- 一般的にコンテンツだけの情報で人気度をモデル化するのは困難

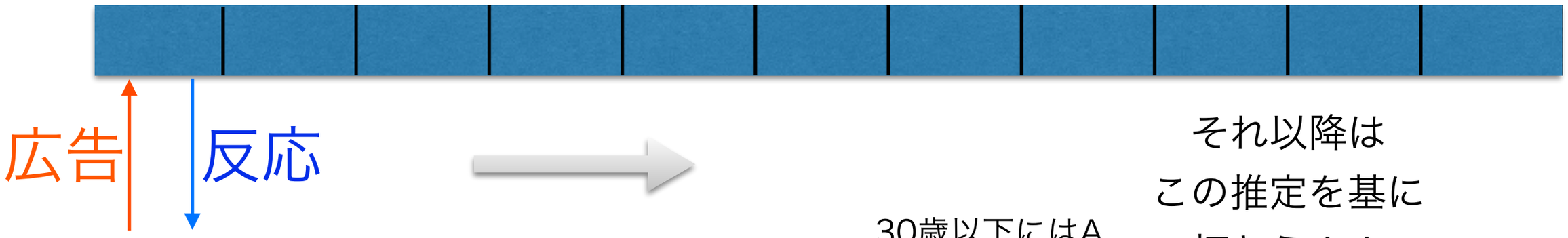
- 実際には、ユーザのフィードバックを集めながら新コンテンツの人気度を評価する

例

- 小さなトラフィックで試して、ユーザの反応を基に残りのトラフィックで人気のものを探していく
    - ε - greedy, EXP3, UCB1,…
    - 新しいコンテンツに対してより多くのトラフィックを掛ける必要はある（当然）

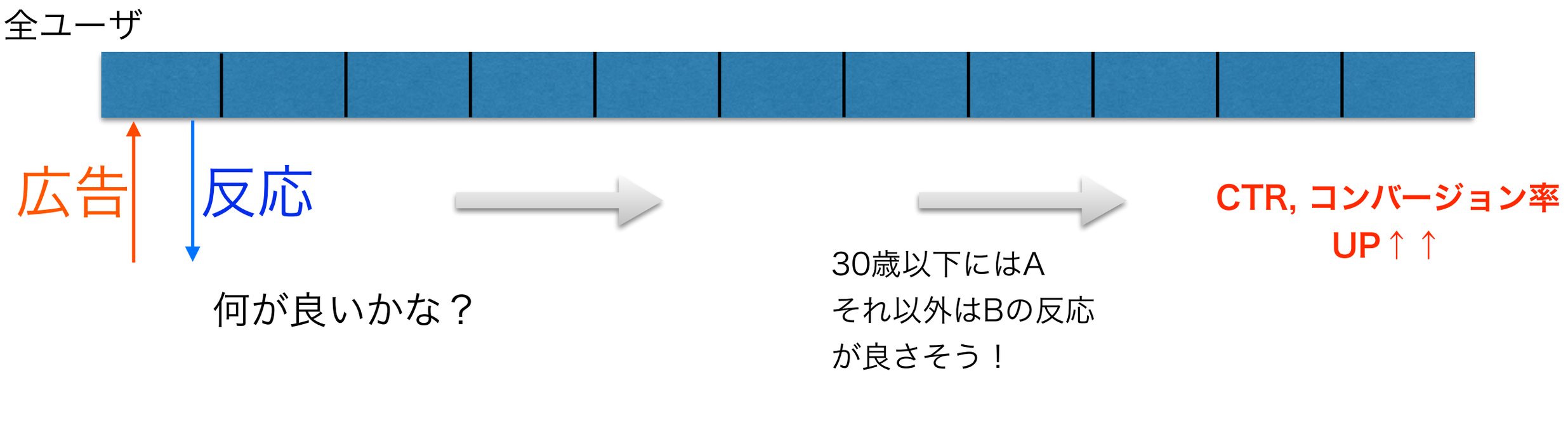
例

- 小さなトラフィックで試して、ユーザの反応を基に残りのトラフィックで人気のものを探していく
    - ε - greedy, EXP3, UCB1,…
    - 新しいコンテンツに対してより多くのトラフィックを掛ける必要はある（当然）



例

- 小さなトラフィックで試して、ユーザの反応を基に残りのトラフィックで人気のものを探していく
    - ε - greedy, EXP3, UCB1,…
- 新しいコンテンツに対してより多くのトラフィックを掛ける必要はある（当然）



協調フィルタリングもあるが

- 多くのWEBサービスでは、コンテンツの多様性は頻繁に変化し、コンテンツの人気は時間とともに変化する
  - cold-startやし無理ゲー。。
- でも新ユーザ、新コンテンツ、もしくは両方成り立つ場合に対して良いマッチングを探すことは必須
  - 現実的に考えると次の課題が生まれる

本研究のモチベーション

- **興味**と**コンテンツ**間の**マッチング情報を集めながら**
- **ユーザ満足度**を長い目で見て**最大化**したい

“最終的に”って意味合い

Contextual Bandit Problemや！

新しいアルゴリズムLinUCB提案したよ！

でも

- インタラクティブだから評価するの難しいお
- **オフラインで前もって性能評価できるような方法**も考えたお！（理論的にもOKだお！）
- **色んなアルゴリズムで評価**したお！

Bandit Algorithm

Contextual-Bandit Algorithm

- Context情報を用いたMulti-armed banditとして
  パーソナライズされた記事推薦の問題をモデル化

- Multi-armed banditは割愛

- 既存研究[1]に従ってContexual banditと呼ぶ

- 便宜上Contextual-Bandit Algorithmを”A”と示す

[1]J.Langford et al., 2008, The epoch-greedy algorithm
for contextual multi-armed bandits

アルゴリズム

- 離散時間の試行をt=1,2,3,…とすると、試行tで

1. “A”は現在のユーザu_t、アームまたは行動の集合A_tを観測。同時にすべての行動aに対し、その特徴ベクトル$x_{t,a}$も観測。$x_{t,a}$はユーザとアーム両方の要約情報であり、”context”と呼ばれる

2. 以前の試行のpayoffに基づき、”A”は行動a_tを選択しpayoff $r_{t,at}$を受け取る。$r_{t,at}$の期待値はユーザu_tとアームa_tの両方に依存する。

アルゴリズム

3. そして新しい観測($x_{t,a}$,a_t,$r_{t,at}$)を使ってアーム選択の戦略を向上させる。ここでは選択されていないアームのフィードバックは観測されないことに注意。

このプロセスを経た、”A”のtotal T-trial payoffは

$$\sum_{t=1}^{T} r_{t,a_t}$$

アルゴリズム

（再掲）”A”のtotal T-trial payoff

$$\sum_{t=1}^{T} r_{t,a_t}$$

同様に、optimal expected T-trial payoffも定義

$$\mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right]$$

a_t^*は試行tでの最大期待payoffをもつアーム

（目標）上の期待値を最大にするように”A”を作る

regretを最小化する

（目標）payoffの期待値を最大にするよう”A”を作る

- “A”のregretを$R_A(T)$と定義すると

実際のpayoff

$$R_{\mathrm{A}}(T) \quad \stackrel{\mathrm{def}}{=} \quad \underset{\text{optimal}}{\mathbf{E}}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbf{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right] \qquad (1)$$

- (目標)はregretを最小化することと等価になる！

重要な特殊ケースを考えておく

- 任意のtで、アーム集合A_tが変化せず、K個のアームをもち、

- ユーザu_t(context($x_{t,1}$,…,$x_{t,K}$))が同じK-armed banditは、

- 普通のbandit algorithmと変わらない

- このContextual banditの特殊ケースは**context-free bandit**とされる

記事推薦のcontext

- 記事をアームのpoolとみなす
- 表示された記事がクリック→payoff=1、他は0
- この定義によりpayoffの期待値はCTRになる
- この研究でのbandit定式化において

CTRを最大化する記事を選択すること

total expected payoffの最大化すること

WEBサービスでは
ユーザ情報獲得が簡単

- 例）若い男どもは退職後の計画よりもiPodの記事が好きだ

- コンパクトに表される有益な情報からユーザと記事を”summarize”

- そうすることでbandit algorithmはユーザ・記事間のCTR情報を汎用的にでき、

- 特に新規ユーザ・記事に対する良い記事の選択を学習できる

探索と搾取のバランス

- bandit問題の基本的な課題は”探索”と”搾取”のバランス
- regretを最小化するために、“A”は最適と思われるアームを過去の経験から**”搾取”**する
- 反対に、もっと情報を得るために準最適なアームも**”探索”**する
- “探索”は短期的regretを増加させるが、長期的regretを減少させる

Contextual-Free Bandit Algorithm

- ε - greedy
  - 1-ε で”搾取”、 ε で”搾取”
  - “unguided”な探索戦略
- UCB
  - “guided”な探索戦略

$$\underbrace{|\hat{\mu}_{t,a}}_{\text{payoffの平均}} - \underbrace{\mu_a|}_{\text{payoff}} < \underbrace{c_{t,a}}_{\text{信頼区間}}$$

$$\underbrace{a_t}_{\text{アーム}} = \arg\max_a \left(\hat{\mu}_{t,a} + c_{t,a}\right)$$

こんなのもあるお

- EXP4($\tilde{O}(\sqrt{T})$)
- epoch-greedy($\tilde{O}(T^{2/3})$)
- LinRel($\tilde{O}(\sqrt{T})$)
- Bayesian approach
- 計算量のオーダーが良いお

提案手法

LinUCB

- UCBアルゴリズムと似たようなものを考案
- LinUCB
  - パラメトリックなpayoff関数が線形
  - 信頼区間が閉形式で効率的に計算可能
  - 一般的なものは”expensive”だから

LinUCB with Disjoint Linear Models

- アームaのexpected payoff

$$\mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_a^*$$

- $x_{t,a}$:d次元のユーザと記事間のcontextベクトル

- θ^*_a:未知の係数ベクトル

d:記事aのcontextの次元

- Disjoint→パラメタがアーム間で共有されていない

広告 CTR = context 20代 * 0.01 + 30代 * 0.02 + 40代以上 * 0.1
係数

LinUCB with Hybrid Linear Models

- mをcontextの数、dをcontextの次元とした時

  計画行列$\mathbf{D}_a$: $\mathbb{R}^{m \times d}$　　ターゲット $\mathbf{b}_a$:$\in \mathbb{R}^m$

- リッジ回帰により係数を推定

cとb間違えてる(?)

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

信頼区間による行動選択

- [2]により、

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

$$\alpha = 1 + \sqrt{\ln(2/\delta)/2}$$

- UCBと同じようにアーム選択を考えることができる

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

expected payoffの標準偏差

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

- リッジ回帰の性質などから$x_{t,a}$の精度向上の寄与度が評価できる
- 上式はpayoff推定とモデル不確実性の削減の間のtrade-offとみなせる

[2]T.J.Walsh et al., 2009, Exploring compact reinforcement-learning representations with linear regression.

LinUCB with Disjoint Linear Models

Algorithm 1 LinUCB with disjoint linear models.

Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d \times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1} \mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
  end for
  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
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t} \mathbf{x}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}$
end for

理論的に定常じゃない入力にも有効らしいお…!

LinUCB with hybrid linear models

- Disjointはアーム固有のfeaturesだけを用いた
- 多くの応用では、アーム間で共有されるfeaturesを用いることは有用→じゃあHybrid modelや

$$\mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{z}_{t,a}^{\top}\boldsymbol{\beta}^{*} + \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_{a}^{*},$$

- $z_{t,a}$:現在のユーザ・記事の組み合わせのfeatures($\mathbb{R}^{k}$)

ちょっとトリッキー(?)

- β:すべての記事で共有される係数

LinUCB with hybrid linear models

- Algorithm 1はもはや使えない
    - 共有featuresが独立でないから
- でも幸運な事にブロック行列の逆行列のテクニック使えばいける！
- 紙面の関係上擬似コードだけ載せるで（詳細はこれからのFull-paperで）

LinUCB with hybrid linear models

リッジ回帰

リッジ回帰

信頼区間計算

Algorithm 2 LinUCB with hybrid linear models.

Inputs: $\alpha \in \mathbb{R}_+$
$\mathbf{A}_0 \leftarrow \mathbf{I}_k$ (k-dimensional identity matrix)
$\mathbf{b}_0 \leftarrow \mathbf{0}_k$ (k-dimensional zero vector)
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $(\mathbf{z}_{t,a}, \mathbf{x}_{t,a}) \in \mathbb{R}^{k+d}$
  $\hat{\boldsymbol{\beta}} \leftarrow \mathbf{A}_0^{-1}\mathbf{b}_0$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{B}_a \leftarrow \mathbf{0}_{d\times k}$ (d-by-k zero matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\left(\mathbf{b}_a - \mathbf{B}_a\hat{\boldsymbol{\beta}}\right)$
    $s_{t,a} \leftarrow \mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{z}_{t,a} - 2\mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{B}_a\mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}$
    $p_{t,a} \leftarrow \mathbf{z}_{t,a}^\top\hat{\boldsymbol{\beta}} + \mathbf{x}_{t,a}^\top\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{s_{t,a}}$
  end for
  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
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{B}_{a_t} \leftarrow \mathbf{B}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{z}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{z}_{t,a_t}\mathbf{z}_{t,a_t}^\top - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + r_t\mathbf{z}_{t,a_t} - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
end for

評価方法

オフラインで評価したい

- でも過去ログは選択された記事だけのログだったり、試したいアルゴリズムと違ったポリシーで貯められてる場合が多いから何も考えないでは使えない
- 過去に配信していない記事は評価できない
- シミュレータを作ってもいいがbiasが入りやすい
- 著者らはログデータを用いてunbiasな方法を提案

評価アルゴリズム

0行目(入力)

T: 得たい履歴数

π: 使用Algorithm

ログ

1,2行目

得た履歴を格納する配列

payoffの初期化

Algorithm 3 Policy_Evaluator.

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

評価アルゴリズム

ログ(例)

1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

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

評価アルゴリズム

t=1

過去の履歴とユーザAの
contextを基に

π

履歴

ログを順番に見ていく

ログ(例)

1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=1

過去の履歴とユーザAの
contextを基に

π → 広告B出す！

実際は広告Aを出してるから
広告Bの反応が見れないので追加しない

履歴

ログ(例)

1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=1

過去の履歴とユーザAの
contextを基に

π

履歴

tはそのままで次のログに行く

ログ(例)
1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=1

過去の履歴とユーザAの
contextを基に

π → 広告B出す！

実際に広告Bを出してるから
履歴に追加する

履歴

ユーザA, 広告B, Non-click

ログ(例)
1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=2

過去の履歴とユーザBの
contextを基に

π → 広告A出す！

実際に広告Aを出してるから
履歴に追加する

履歴

ユーザA, 広告B, Non-click
ユーザB, 広告A, Click

追加したらt+1

ログ(例)
1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=3

過去の履歴とユーザCの
contextを基に

π → 広告A出す！

実際に広告Aを出してるから
履歴に追加する

履歴

ユーザA, 広告B, Non-click
ユーザB, 広告A, Click
ユーザC, 広告A, Non-click

ログ(例)
1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click

評価アルゴリズム

t=100

最終的な履歴を見て

CTR = 1/3 = 33.3%

履歴

ユーザA, 広告B, Non-click
ユーザB, 広告A, Click
ユーザC, 広告A, Non-click

ログ(例)
1/1,ユーザA, 広告A, Click
1/1,ユーザA, 広告B, Non-click
1/1,ユーザB, 広告A, Click
1/1,ユーザC, 広告A, Non-click
1/1,ユーザD, 広告B, Non-click
1/1,ユーザC, 広告B, Non-click
1/1,ユーザA, 広告A, Click
1/1,ユーザB, 広告B, Non-click
…

数学的な証明

- (定理)長さTの履歴を得るために必要なログデータの期待値はKT
- Kはアーム、記事の数
- 証明されてる。

評価方法の感想

賢いやん

実験

setup

Yahoo! Today-Module

ここに何の広告を
出すか決める

F1に出される広告は
STORYにでかく表示される

Data Collection

- Tuning data: 1/May/2009
    - 最適なパラメタ決定に使用
        - 4.7 million events
- Evaluation data: 03-09/May/2009
    - 36 million events

Data Collection

- ユーザのインタラクションイベントは3つ
  - ランダムに選択された記事
  - ユーザ・記事の情報
  - ユーザがSTORYにあるその広告をクリックしたか

Feature Construction

- 0.1以上の“支持度”をもつ特徴量を抜き出す
- 各**ユーザ**は1000以上のraw featuresで表される
  - demographic: 性別、年齢（10個にセグメント）
  - geographic: 世界の200都市、U.S.州
  - behavioral: Yahoo!でのユーザの消費をまとめた1000バイナリカテゴリ

Feature Construction

- 各**記事**は約100個のfeatureをもつ
  - URL カテゴリ
  - 編集カテゴリ: エディタによるタグ付けトピック
- [3]によりfeatureをbinaryにencode
- 最後[0, 1]にnormalize
- ☆　値が1のconstant featureも追加

重要そう（感想）

[3]W. Chuet al., 2009, Personalized recommendation on dynamic content using predictive bilinear models

Feature Reduction

- 現実的には小さな特徴空間でやらないと辛いだろう
- ということで[4]使って、ユーザ特徴量を記事カテゴリに射影して、K-means使ってクラスタ化した
  - コンジョイント分析
- ユーザと記事をそれぞれ５個にセグメント

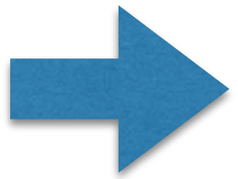
5つのユーザセグメント

$$u \begin{bmatrix} 0.1 & 0.9 & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.3 & 0.3 & 0.2 & 0.1 \\ \cdots & & & & \end{bmatrix}$$

5つの記事セグメント

$$a \begin{bmatrix} 0.0 & 0.4 & 0.0 & 0.4 & 0.1 \\ 0.9 & 0.0 & 0.0 & 0.1 & 0.0 \\ \cdots & & & & \end{bmatrix}$$

これに1ベクトル
足して外積



$z_{t,a}$
共有特徴量

比較するアルゴリズム

比較するアルゴリズム

1. Context-freeなAlgorithm
    - random
    - ε - greedy
    - UCB
    - omniscient(後知恵から得る過去一番のCTR記事)
2. Algorithm with “warm-start”
    - Context-freeなアルゴにユーザ固有のCTR情報を足し合わせる
3. オンラインなユーザ固有のAlgorithm
    - ε - greedy, UCB (セグメント分けされたユーザ使用)
    - ε - greedy, UCB (disjoint & hybrid)

Performance Metric

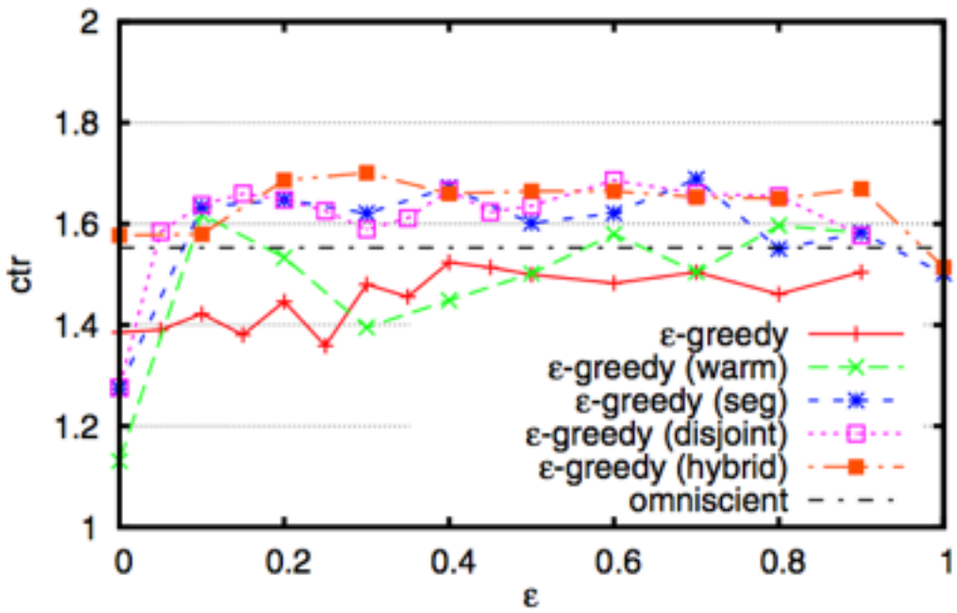
Performance Metric

- ページへの全トラフィックをランダムに2つに分割
  1. learning bucket : アルゴリズム学習用
  2. deployment bucket : アルゴリズム実験用

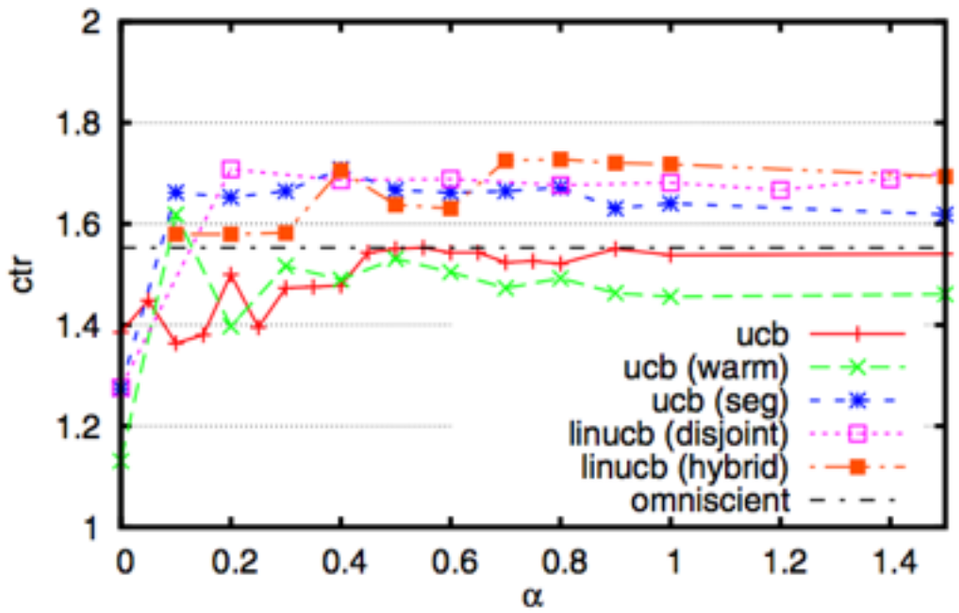
- deployの方が重要だから大きなbucketになりがち

- だけど、learningもより速い学習(より小さなregret)を示唆するので両方の結果を示す

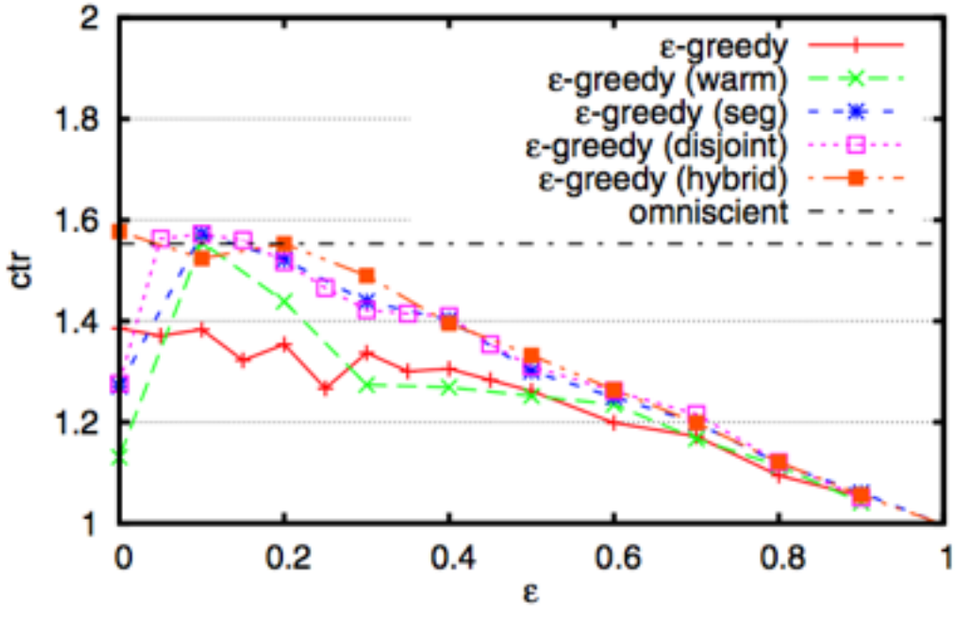
チューニングデータでの結果



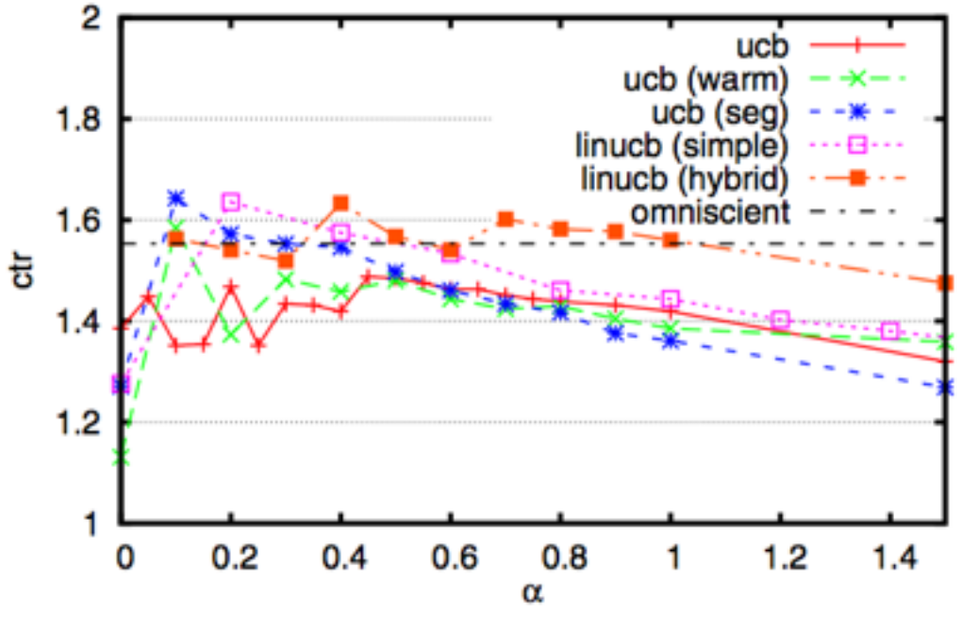
(a) Deployment bucket.



(b) Deployment bucket.



(c) Learning bucket.



(d) Learning bucket.

全オンライン学習アルゴリズムはOmniscentに勝ってる

評価データでの結果

全データ

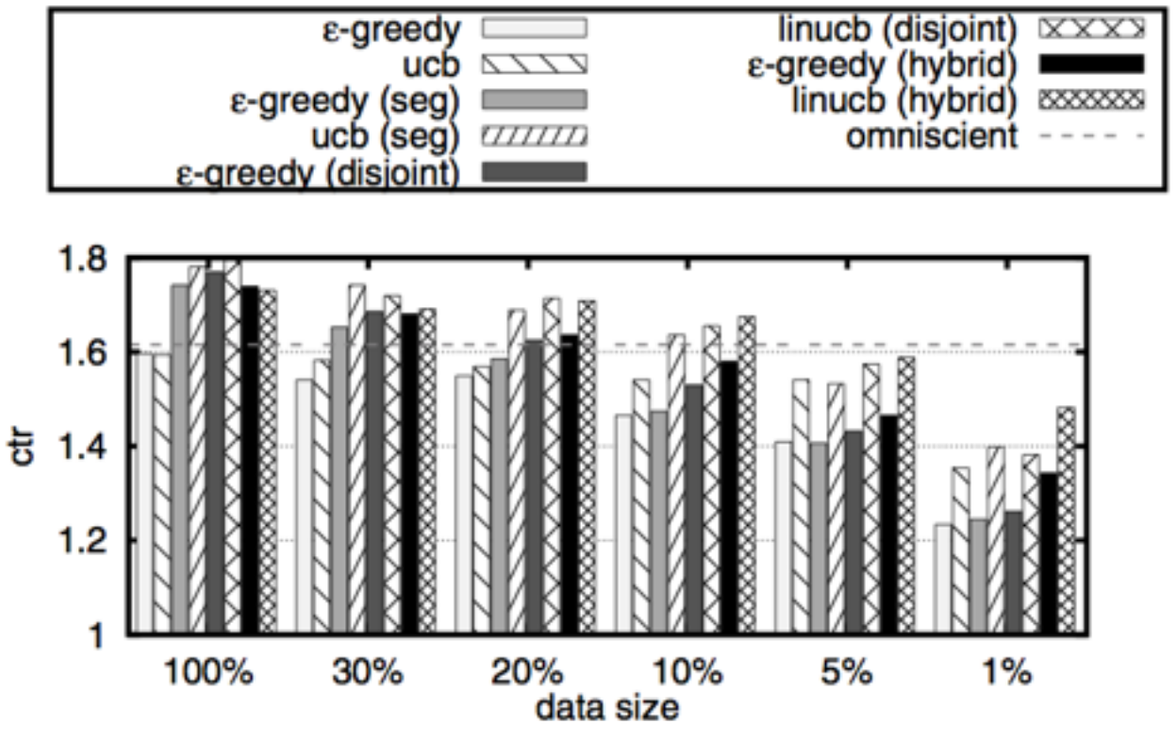
よりスパースなデータ

algorithm	size = 100%		size = 30%		size = 20%		size = 10%		size = 5%		size = 1%	
	deploy	learn	deploy	learn	deploy	learn	deploy	learn	deploy	learn	deploy	learn
ϵ-greedy	1.596 0%	1.326 0%	1.541 0%	1.326 0%	1.549 0%	1.273 0%	1.465 0%	1.326 0%	1.409 0%	1.292 0%	1.234 0%	1.139 0%
ucb	1.594 0%	1.569 18.3%	1.582 2.7%	1.535 15.8%	1.569 1.3%	1.488 16.9%	1.541 5.2%	1.446 9%	1.541 9.4%	1.465 13.4%	1.354 9.7%	1.22 7.1%
ϵ-greedy (seg)	1.742 9.1%	1.446 9%	1.652 7.2%	1.46 10.1%	1.585 2.3%	1.119 −12%	1.474 0.6%	1.284 −3.1%	1.407 0%	1.281 −0.8%	1.245 0.9%	1.072 −5.8%
ucb (seg)	1.781 11.6%	1.677 26.5%	1.742 13%	1.555 17.3%	1.689 9%	1.446 13.6%	1.636 11.7%	1.529 15.3%	1.532 8.7%	1.32 2.2%	1.398 13.3%	1.25 9.7%
ϵ-greedy (disjoint)	1.769 10.8%	1.309 −1.2%	1.686 9.4%	1.337 0.8%	1.624 4.8%	1.529 20.1%	1.529 4.4%	1.451 9.4%	1.432 1.6%	1.345 4.1%	1.262 2.3%	1.183 3.9%
linucb (disjoint)	1.795 12.5%	1.647 24.2%	1.719 11.6%	1.507 13.7%	1.714 10.7%	1.384 8.7%	1.655 13%	1.387 4.6%	1.574 11.7%	1.245 −3.5%	1.382 12%	1.197 5.1%
ϵ-greedy (hybrid)	1.739 9%	1.521 14.7%	1.68 9%	1.345 1.4%	1.636 5.6%	1.449 13.8%	1.58 7.8%	1.348 1.7%	1.465 4%	1.415 9.5%	1.342 8.8%	1.2 5.4%
linucb (hybrid)	1.73 8.4%	1.663 25.4%	1.691 9.7%	1.591 20%	1.708 10.3%	1.619 27.2%	1.675 14.3%	1.535 15.8%	1.588 12.7%	1.507 16.6%	1.482 20.1%	1.446 27%

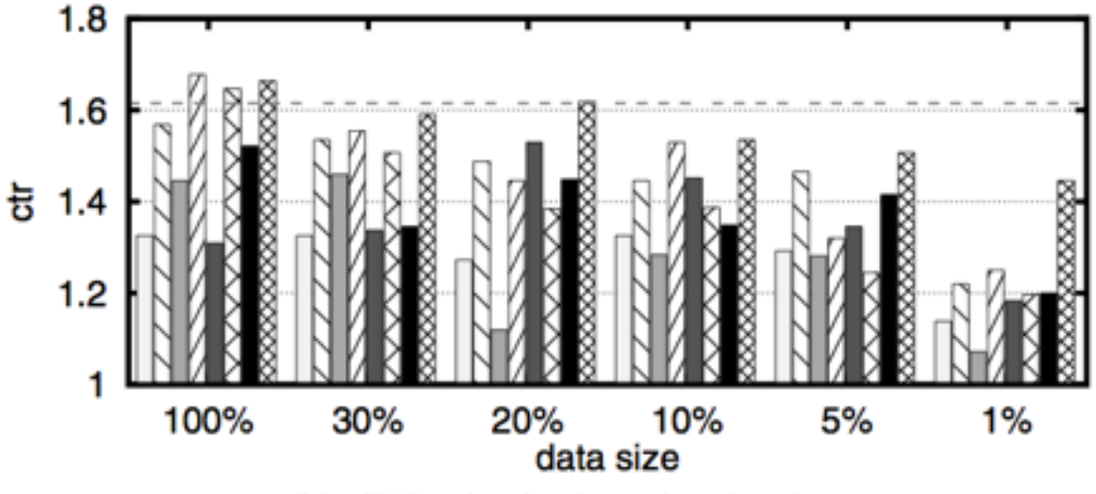
random: 1,　omniscient: 1.615

スパースなデータ

- 下がり気味
- UCB系強い
- hybridは記事間の情報を共有してるから強い

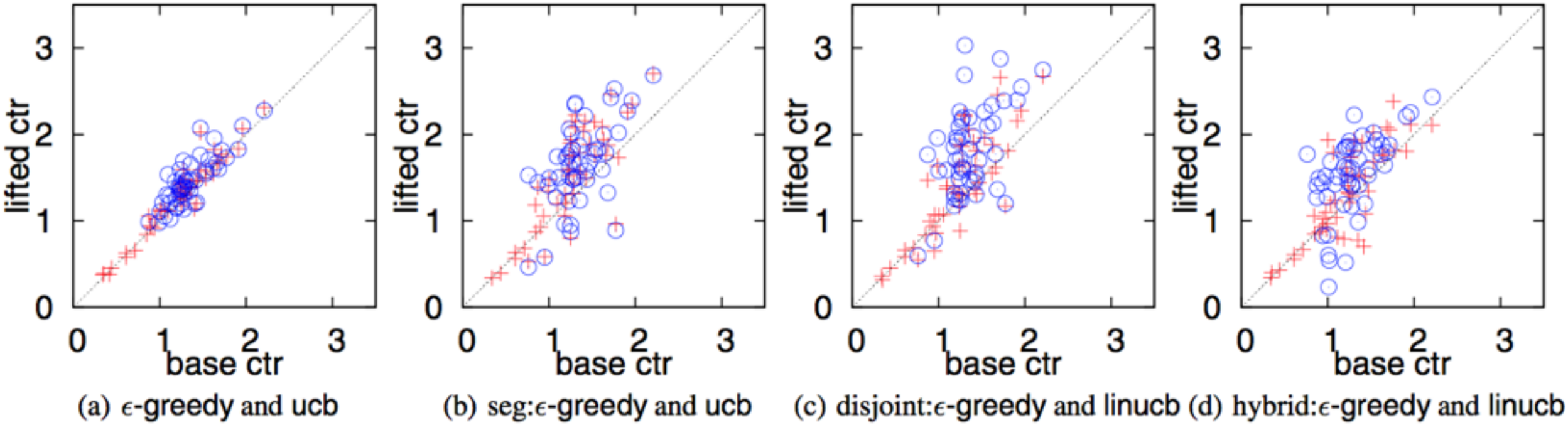


(a) CTRs in the deployment bucket.



(b) CTRs in the learning bucket.

Context-freeとContextの比較



(a) ϵ-greedy and ucb (b) seg:ϵ-greedy and ucb (c) disjoint:ϵ-greedy and linucb (d) hybrid:ϵ-greedy and linucb

- 評価データの中で最も選ばれた50個の記事を使用
- Personalized recommendationは有効
- Base ctrはcontext-freeなそのアルゴリズム

一応ユーザの分割も良かたよ

- 各ユーザのメンバシップの最大値のヒストグラム
- 85%が0.5以上、40%が0.8以上

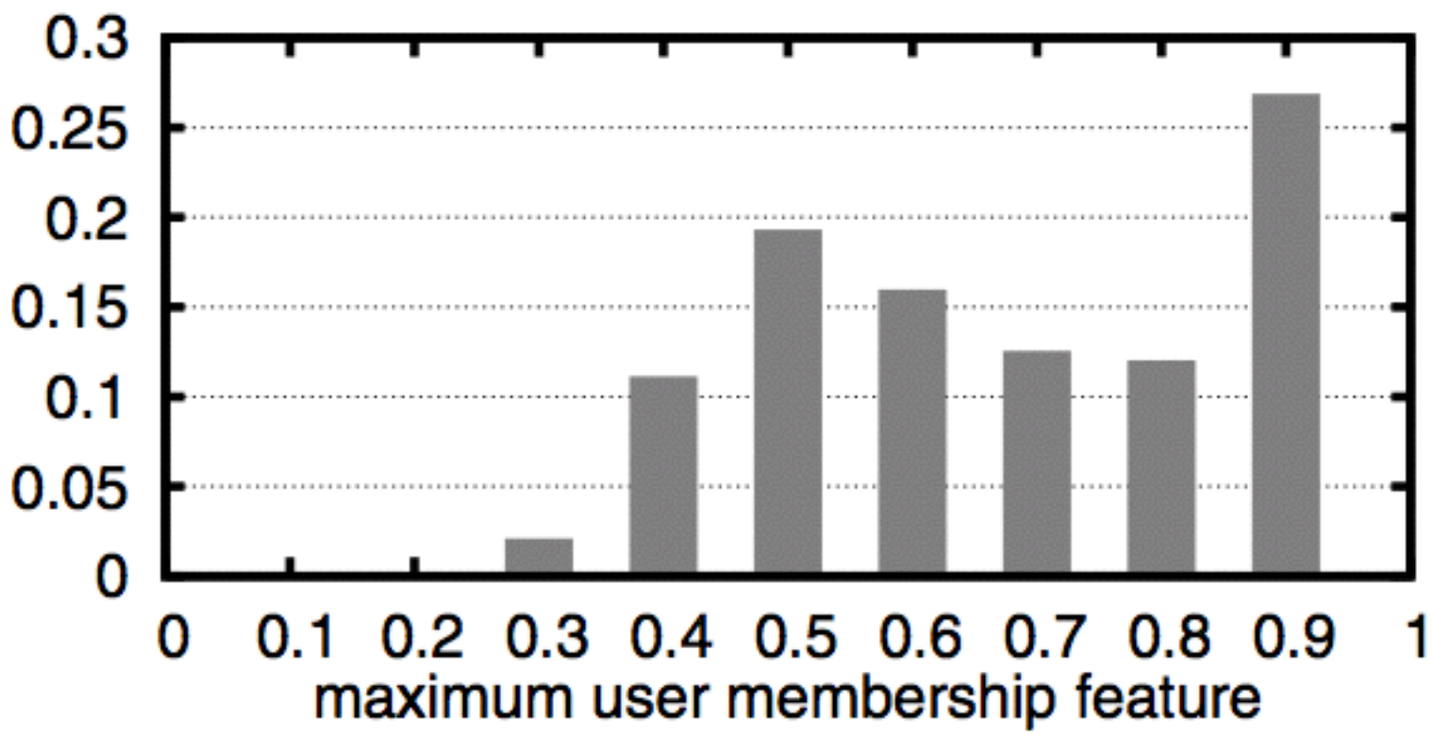


Figure 5: User maximum membership histogram.

結論

結論

- contextual-banditな方法をニュースレコメンドのようなパーソナライズされたWEBサービスに適用した

- LinUCBとシンプルで信頼性のある評価方法提案した

参考文献

- Lihong Li et al., 2010, A Contextual-Bandit Approach to Personalized News Article Recommendation
- http://www.slideshare.net/tsubosaka/contexual-bandit