

効果のある クリエイティブ広告の 見つけ方

自己紹介

- サイバー何ちらの新卒1年目
- 配属の結果、CTRひたすら予測するマンにならないといけないので色々勉強中
研修ではGo触ったりしてましたが今はPythonメイン
- twitter:@coldstart_p(前のは恥ずかしくなって変えました)
github:<https://github.com/ykaneko1992> (今日のシミュレーションのコードをあげます)

今日の話

Contextual Banditにおいて、
決定的アルゴリズムのUCBよりベイズ的アプローチのThompson
Samplingの方がよくなる、というお話

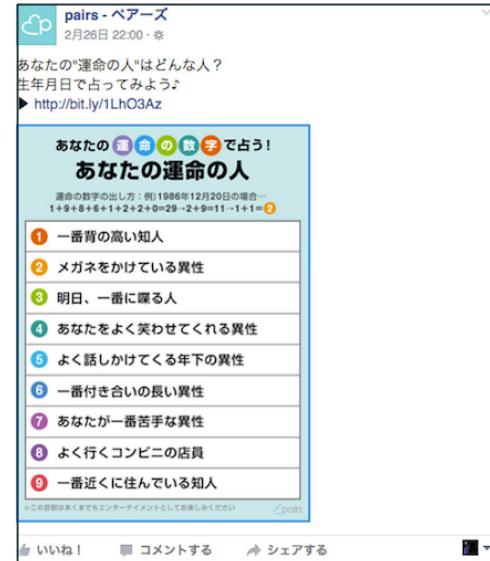
元ネタ

- 元ネタ: [Dimakopoulou, Maria, Susan Athey, and Guido Imbens. "Estimation Considerations in Contextual Bandits." arXiv preprint arXiv:1711.07077 \(2017\).](#)
→ 元々はContextual Banditは推定法に対して sensitiveだというモチベーションから、パラメトリックな手法とノンパラメトリックな手法での推定法を提案した論文
- 全部紹介していると時間がないので、パラメトリックな手法のみの紹介に絞ります
→ generalized random forestを使用したノンパラ推定や、Inversed Propensity Score(IPW)を用いた補正などが紹介されているので興味があれば元論文を読んでください

どっちの広告を出すべきか？



広告A
CTR:15%
クリック数:1500
imp数: 10000



広告B
CTR:10%
クリック数:10
imp数:100

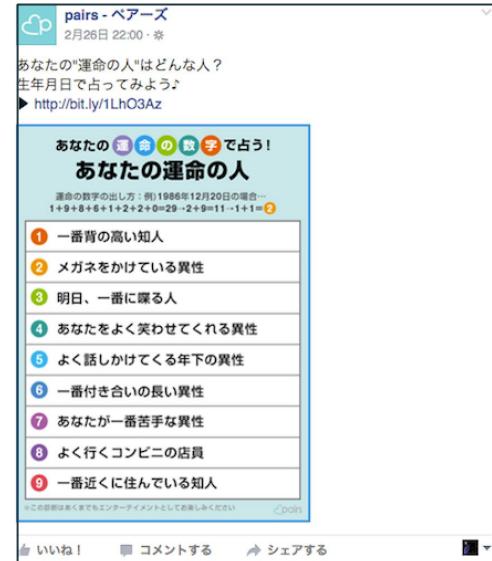
問題：
次のユーザーにどちらの広告を配信すべきか？
→Bの方がいい可能性もある
→どれくらいimp数重ねればいい？
試行の中で利益得るには？

バンディット(Bandit)アルゴリズム

- モチベ:「どちらの広告を配信すべきか」という意思決定を N回繰り返して、最終的に多くのクリック数を得たい
- Aだけ配信してると、実はBの方がよかつた場合には損
→Bが本当はどんなCTRなのかを確認しつつ、適度にAでクリックを稼ぎたい！
- 本当のCTRの確認に極振りするとA/Bテストになるが、要はこれも損をしてる
- クリック率の期待値と分散を用いて、上のことを行うのが **バンディットアルゴリズム**
- 詳しく知りたい人は本多・中村本参照



どっちの広告を出すべきか？(続)



広告A
CTR:15%
imp数:20000
CTR(男女別)
男性:25% 女性: 2%

広告A
CTR:12%
imp数:10000
CTR(男女別)
男性:1% 女性: 20%

問題:

次のユーザーにどちらの広告を配信すべきか？

→男女の差異が非常に大きい

→通常のBanditだと男女差を考慮できない

Contextual Bandit

- クリック率の期待値と分散を、特定の変数(コンテクスト、今回は男女)ごとに変えれば、男女で取る選択肢を変えることができる
→**Contextual Bandit**
- 具体的には、変数ごとに回帰モデルを持つなどして実行する

どうやって選択肢を選ぶのか

- 以下の2つの方法(本田・中村 3章)

1.UCB アルゴリズム

→信頼区間の上限 (Upper Confidence Bound :UCB) が最大になる選択肢を引く
= 決定論的アルゴリズム

2.Thompson Sampling(TS)

→その選択肢が最適である事後確率に従って選択肢をランダムに選択
=ランダムアルゴリズム(ベイズ戦略)

- Thompson Samplingの方がいい！ という話をします

論文のシミュレーション設定

- 選択肢3つ(0,1,2), コンテクスト2つ(X_{i0} , X_{i1}), 選択肢の利得は以下の図 (グラフの縦軸は報酬の期待値)

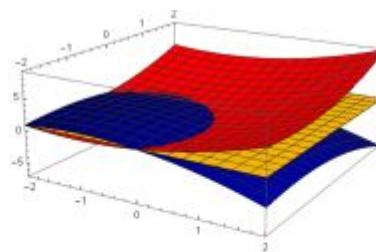


Figure 1: Expectation of potential outcomes, $\mathbb{E}[Y_i(0)] = 0.5(X_{i0} + 1)^2 + 0.5(X_{i1} + 1)^2$ (red), $\mathbb{E}[Y_i(1)] = 1$ (yellow), $\mathbb{E}[Y_i(2)] = 2 - 0.5(X_{i0} + 1)^2 - 0.5(X_{i1} + 1)^2$ (blue).

- 利得の式に1次項と2次項が入っていることに注意

手法

1. コンテクストを50セット用意して、一様ランダムに選択肢を引く(コールドスタート)
2. 各選択肢ごとにブートストラップを用いて復元抽出をして(今回は100回)リッジ回帰を回す
これによって、各選択肢ごとに標本分布の平均と分散が得られる
3. 平均、分散を元に次の10セットを以下の基準で選択をする(バッチ学習)

UCB:

$$W_i = \operatorname{argmax}_w \{ \hat{\mu}_w(\mathbf{x}) + \beta \sqrt{2 \log t} \hat{\sigma}_w(\mathbf{x}) \},$$

となる選択肢を選ぶ

TS: $\mathcal{N}(\hat{\mu}_w(X_i), \hat{\sigma}_w^2(X_i))$.

に従い各選択肢で乱数を生成し、乱数が最大になった選択肢を選ぶ

4. 各バッチごとに10セットのコンテクストを追加して2,3のプロセスを繰り返す.
 - 今回はパラメトリックなのでリッジ回帰をまわす時は1次項と2次項を含めて回す

シミュレーション

- Rでシミュレーションを回してみた
- 下の図(再掲)を上から見て2次元化したものを再現する
つまり、右上に赤(0), 左下に青(2)が来るようになればうまくいっていると言える

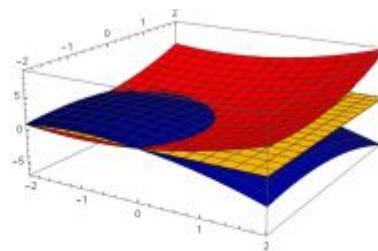
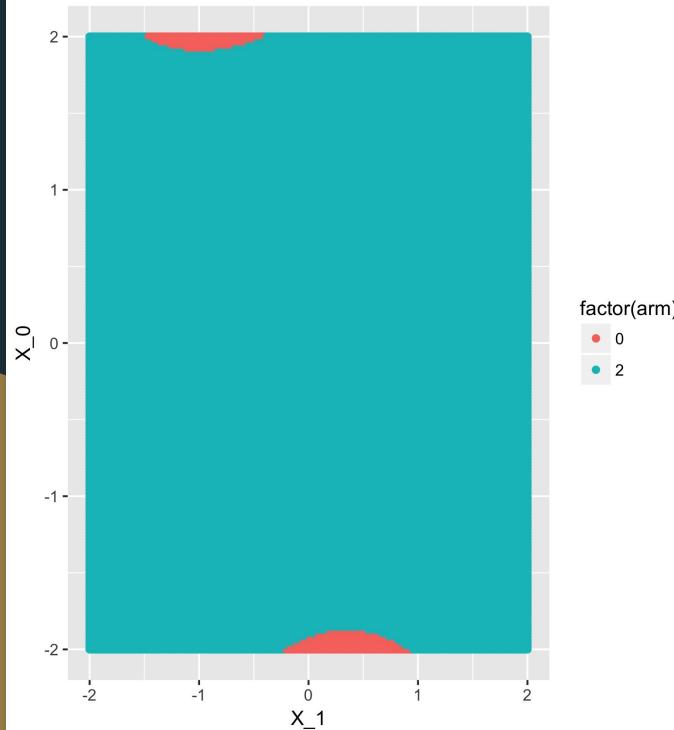


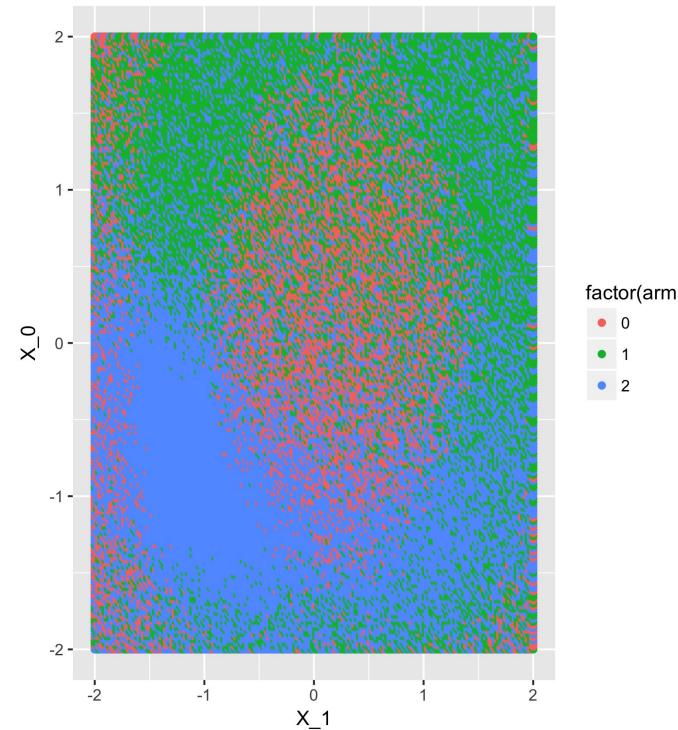
Figure 1: Expectation of potential outcomes, $\mathbb{E}[Y_i(0)] = 0.5(X_{i0} + 1)^2 + 0.5(X_{i1} + 1)^2$ (red),
 $\mathbb{E}[Y_i(1)] = 1$ (yellow), $\mathbb{E}[Y_i(2)] = 2 - 0.5(X_{i0} + 1)^2 - 0.5(X_{i1} + 1)^2$ (blue).

シミュレーション(左がUCB, 右がTS)

UCB Batch 1

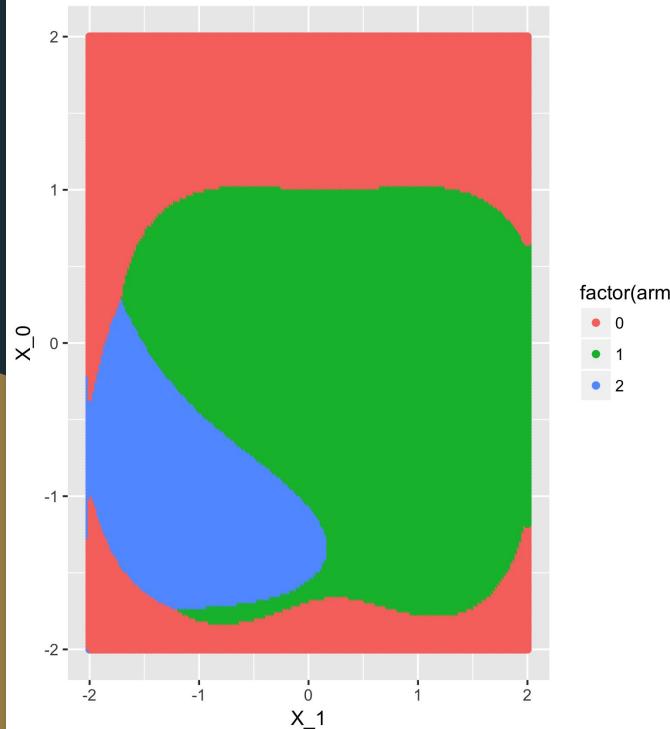


TS Batch 1

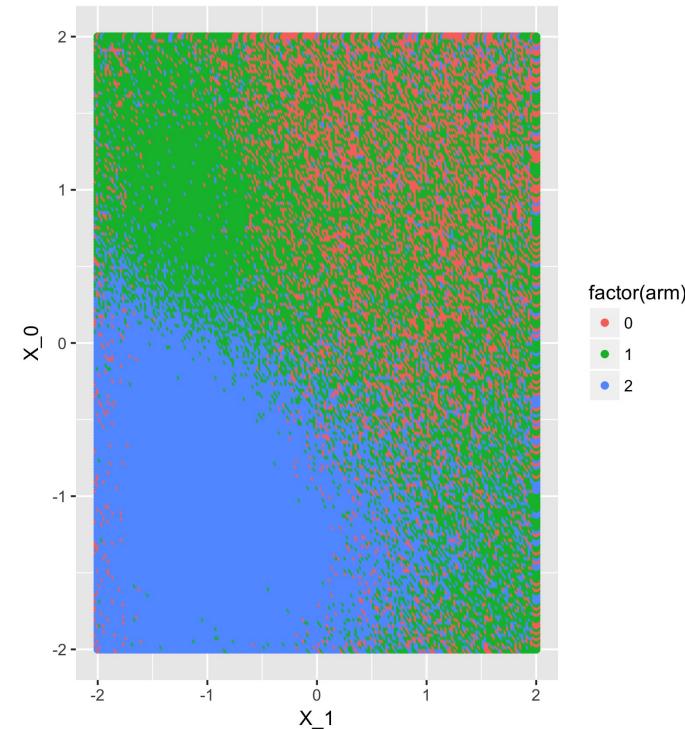


シミュレーション(左がUCB, 右がTS)

UCB Batch 2

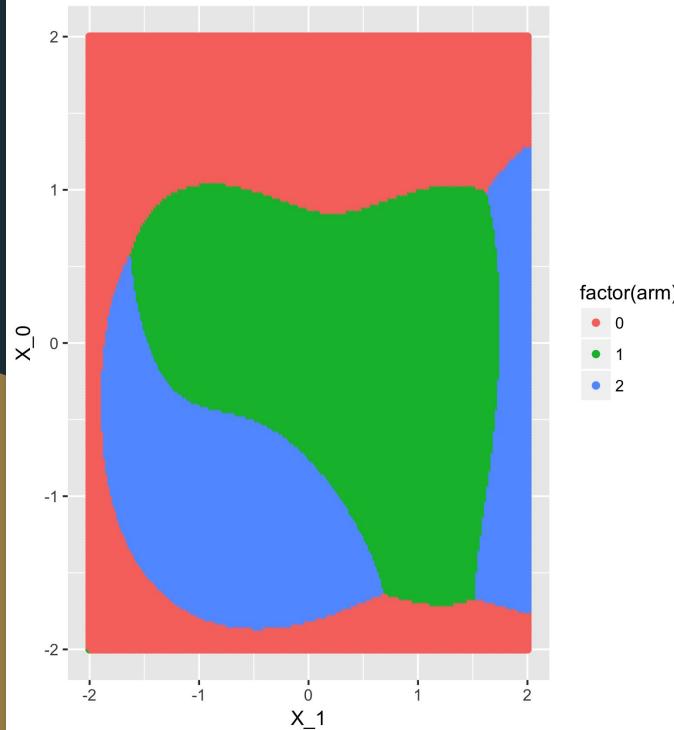


TS Batch 2

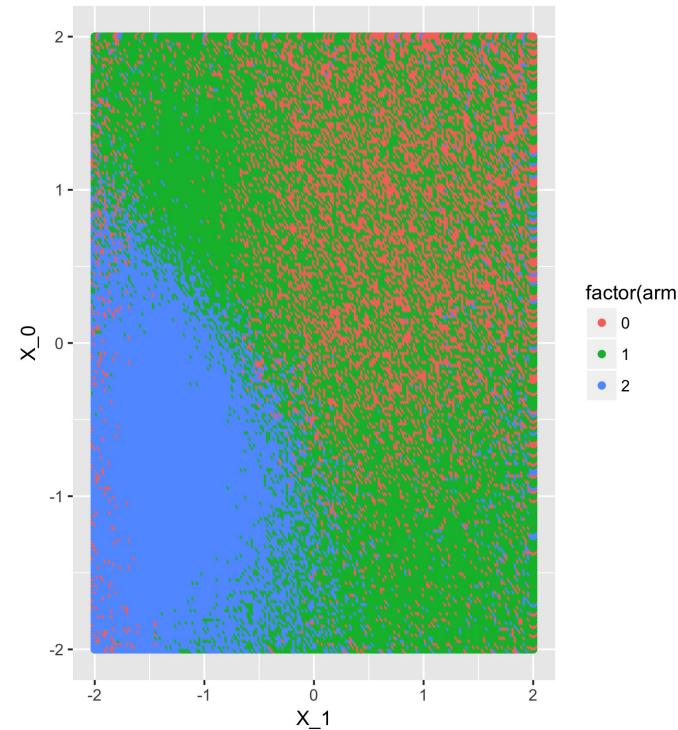


シミュレーション(左がUCB, 右がTS)

UCB Batch 3

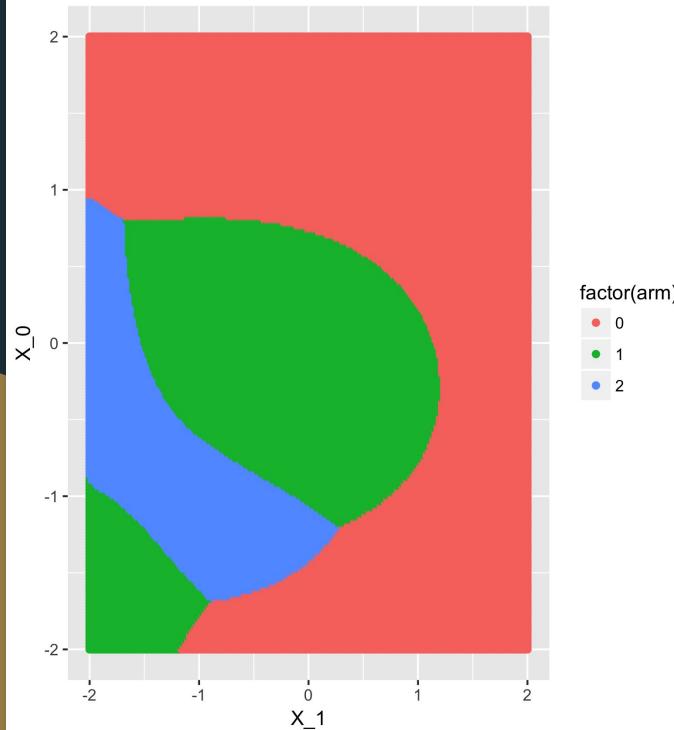


TS Batch 3

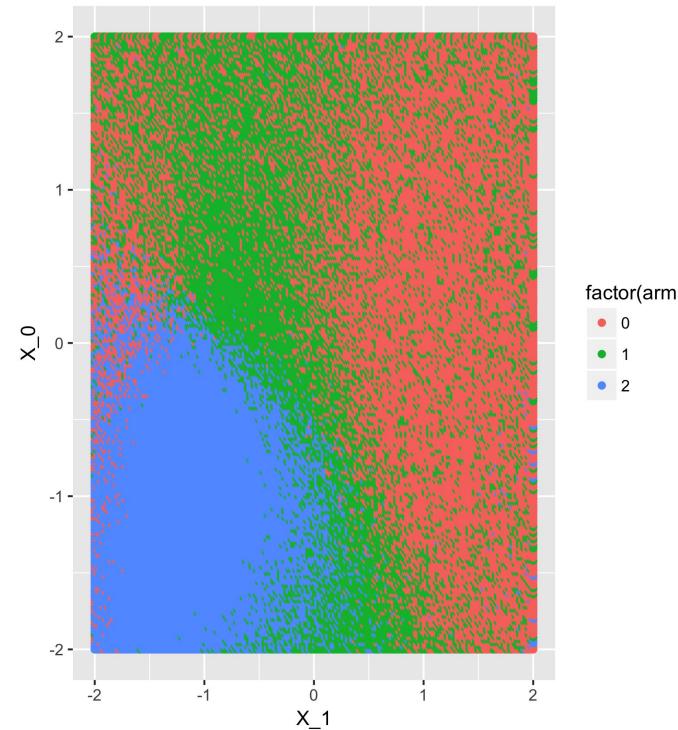


シミュレーション(左がUCB, 右がTS)

UCB Batch 4

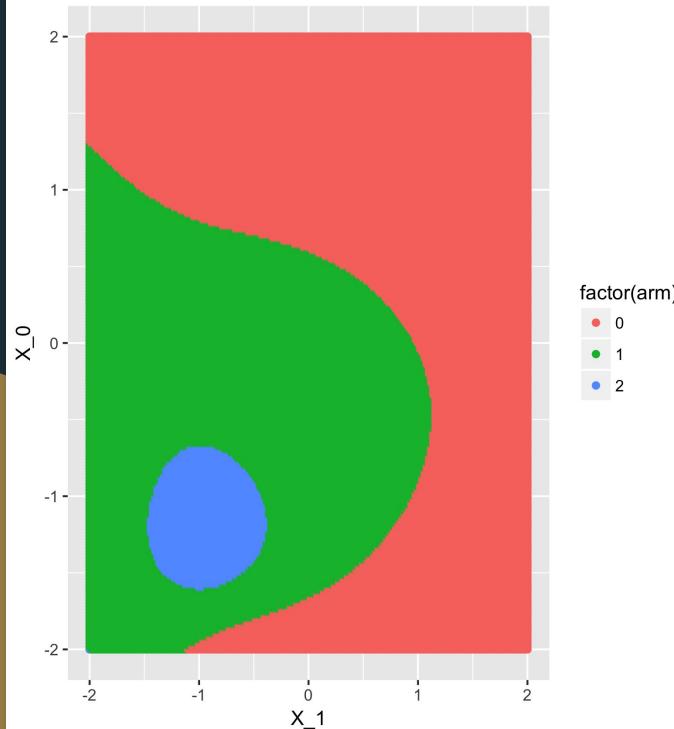


TS Batch 4

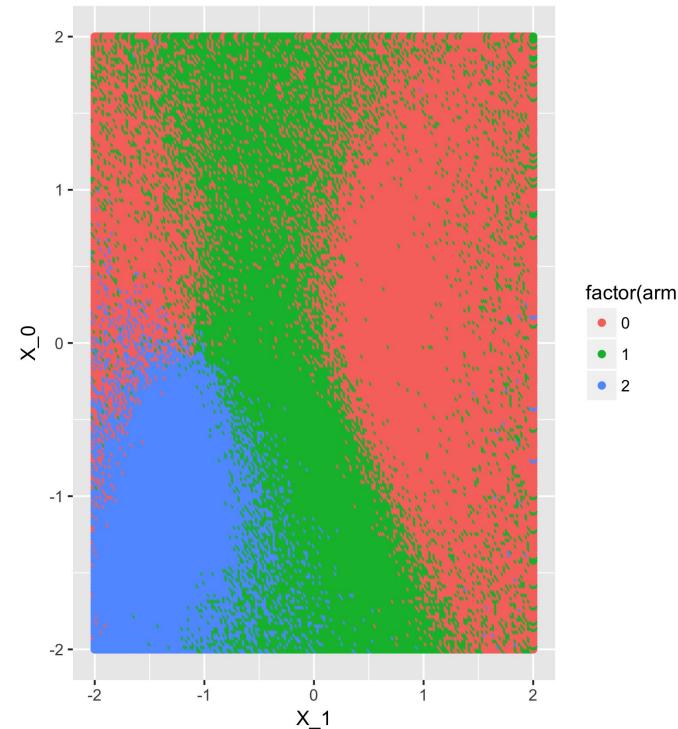


シミュレーション(左がUCB, 右がTS)

UCB Batch 5

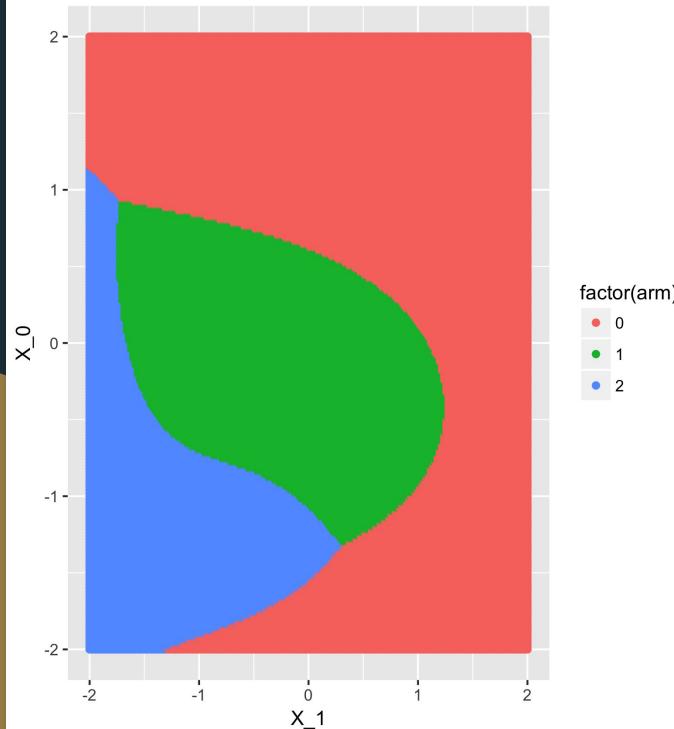


TS Batch 5

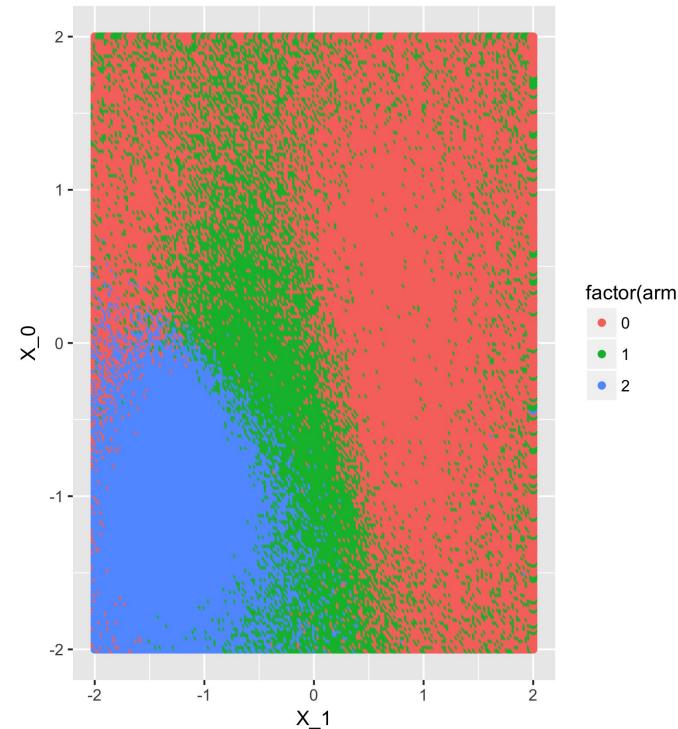


シミュレーション(左がUCB, 右がTS)

UCB Batch 6

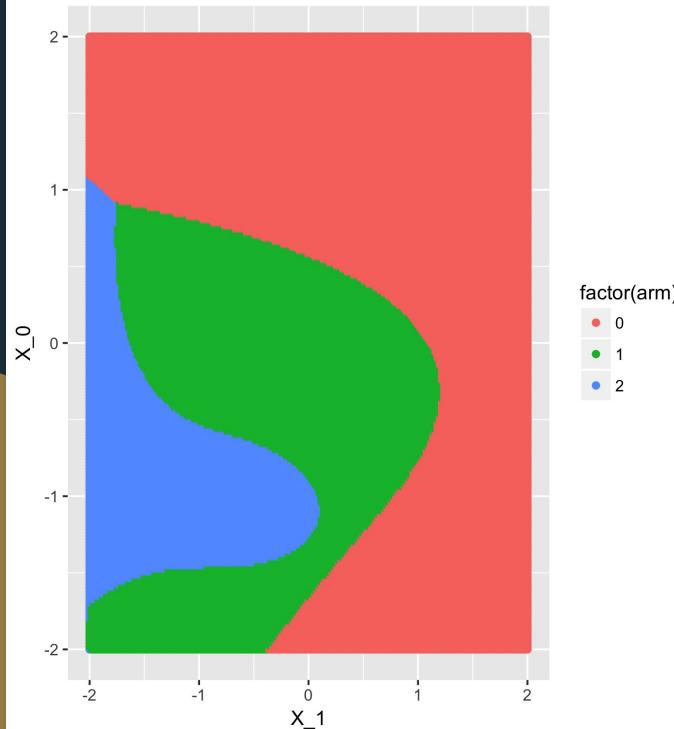


TS Batch 6

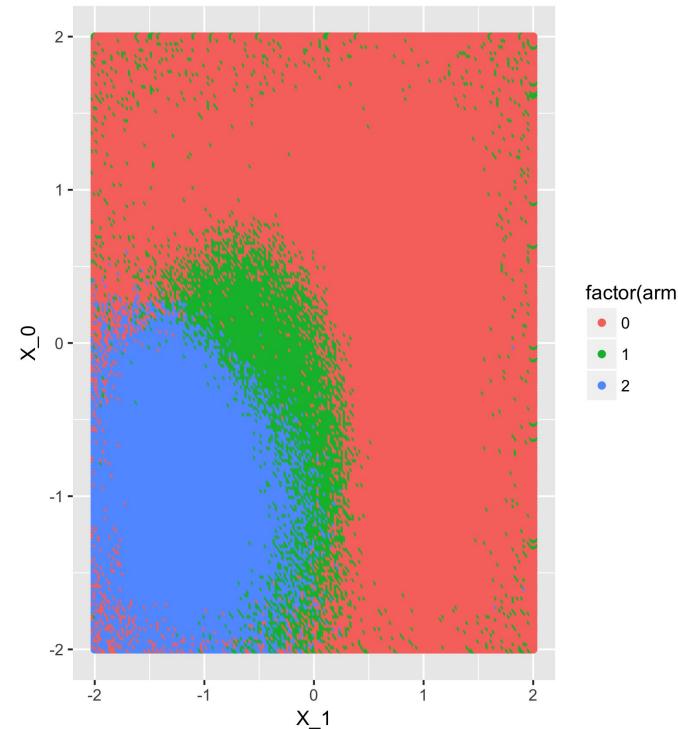


シミュレーション(左がUCB, 右がTS)

UCB Batch 7

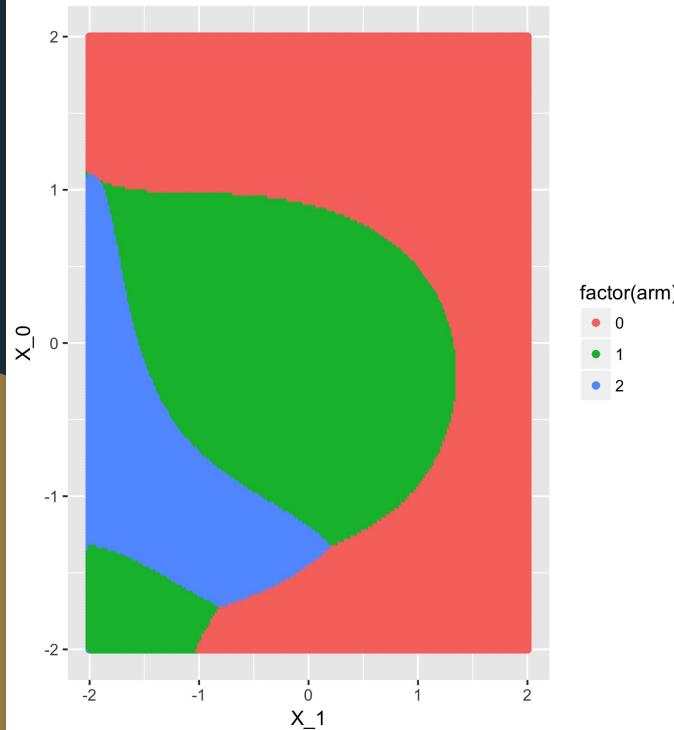


TS Batch 7

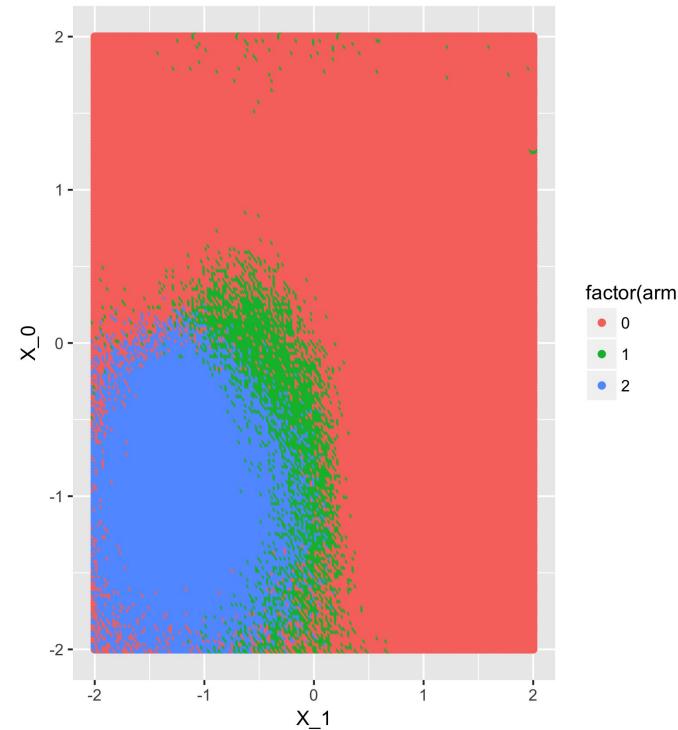


シミュレーション(左がUCB, 右がTS)

UCB Batch 8

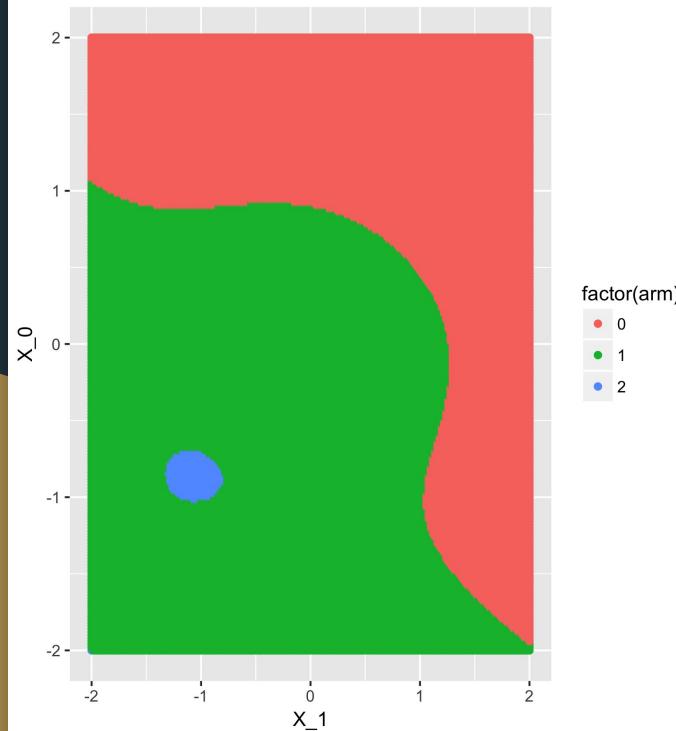


TS Batch 8

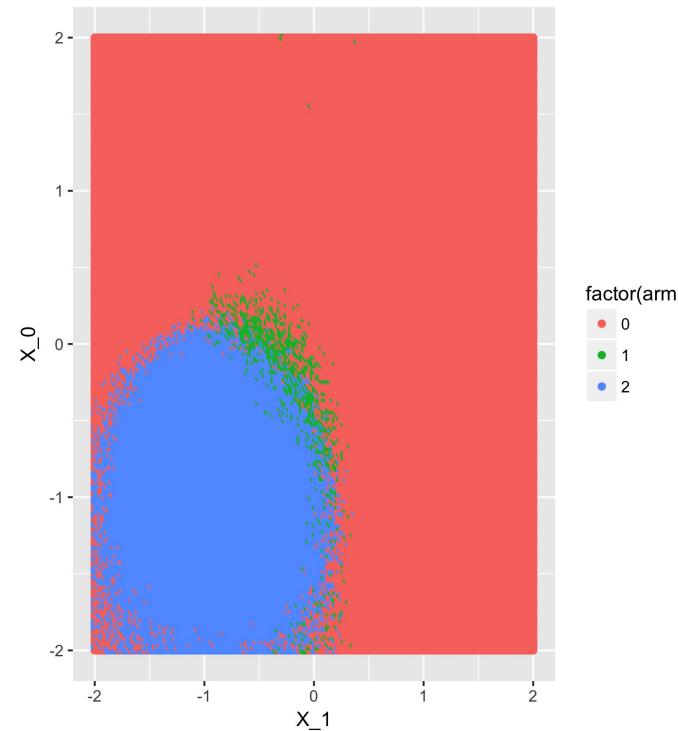


シミュレーション(左がUCB, 右がTS)

UCB Batch 9

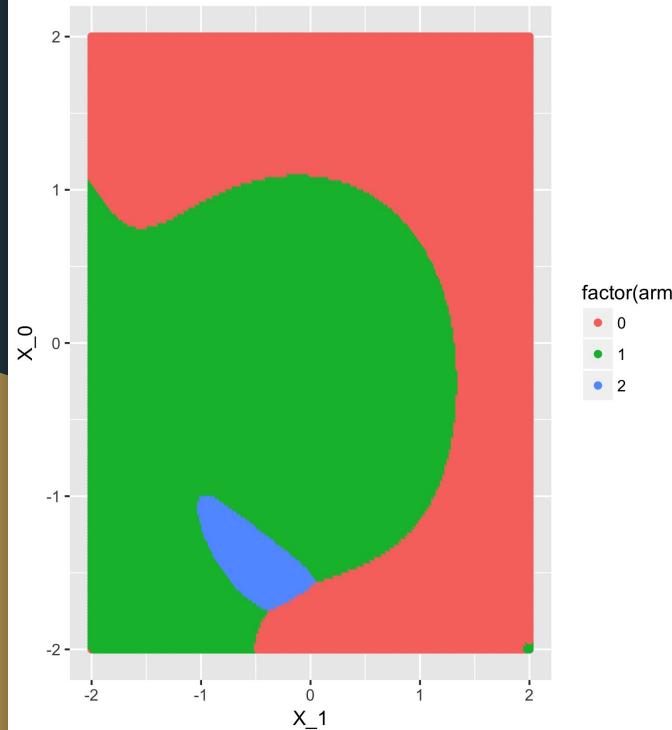


TS Batch 9

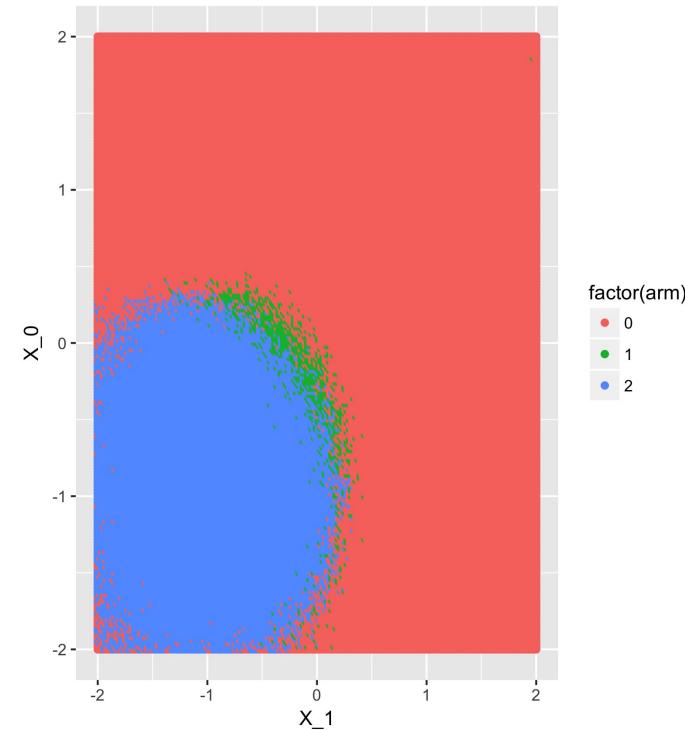


シミュレーション(左がUCB, 右がTS)

UCB Batch 10

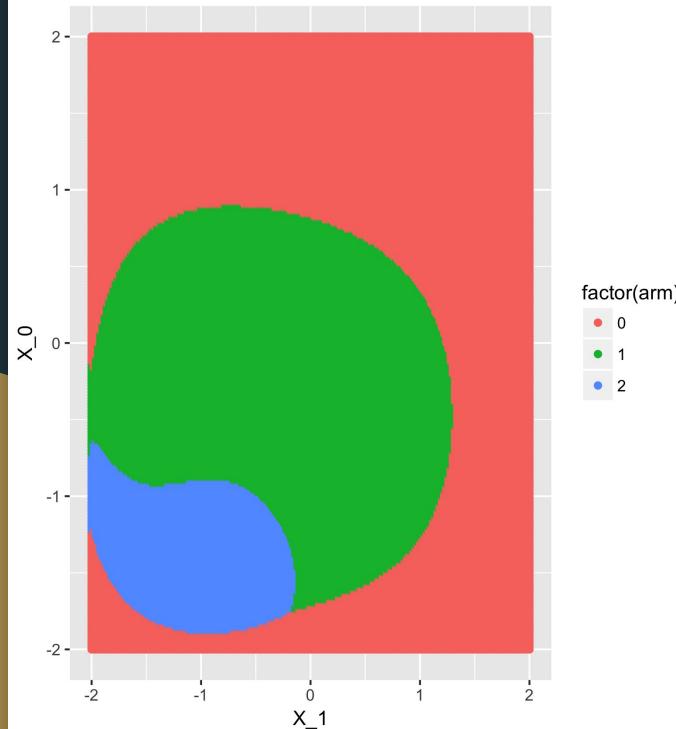


TS Batch 10

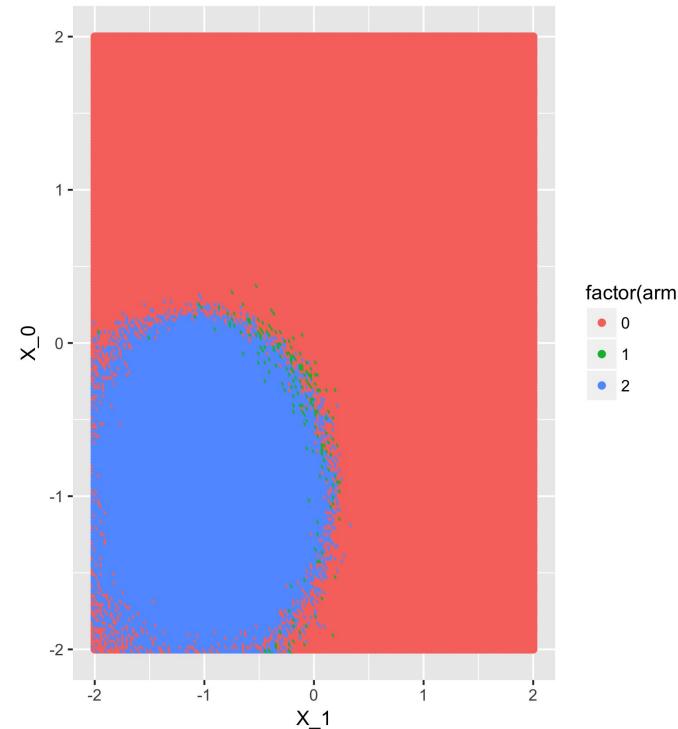


シミュレーション(左がUCB, 右がTS)

UCB Batch 11

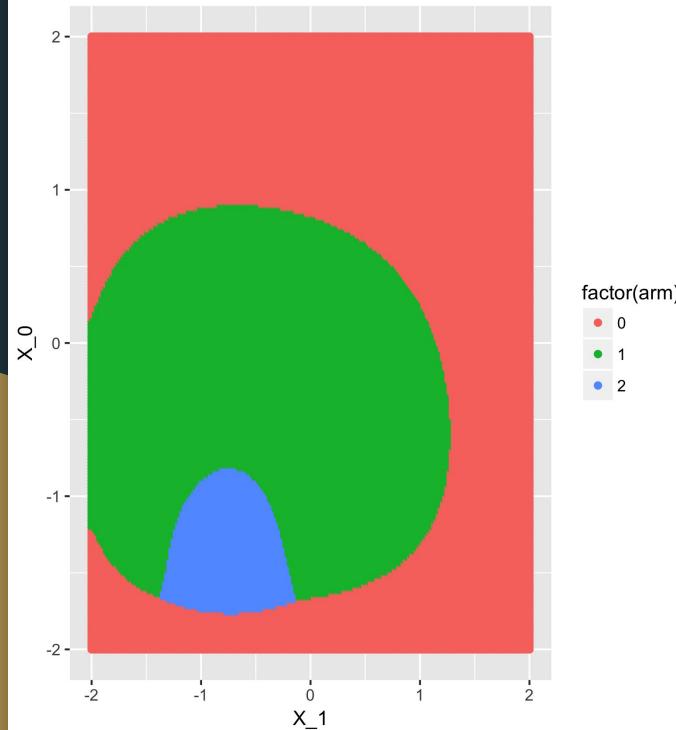


TS Batch 11

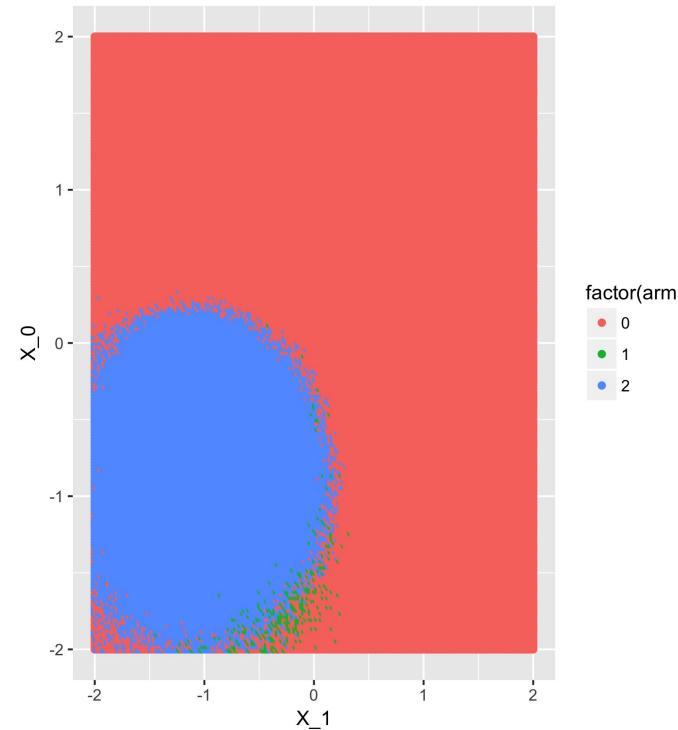


シミュレーション(左がUCB, 右がTS)

UCB Batch 12

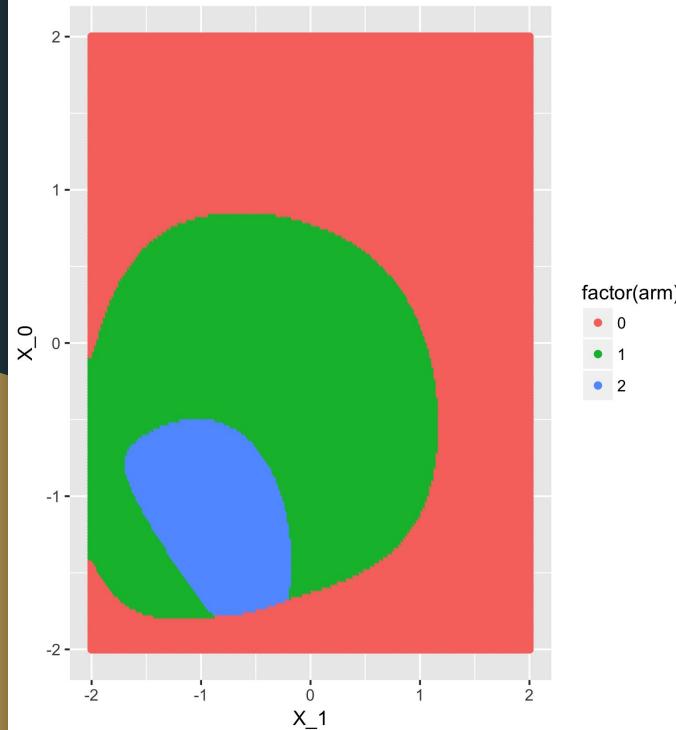


TS Batch 12

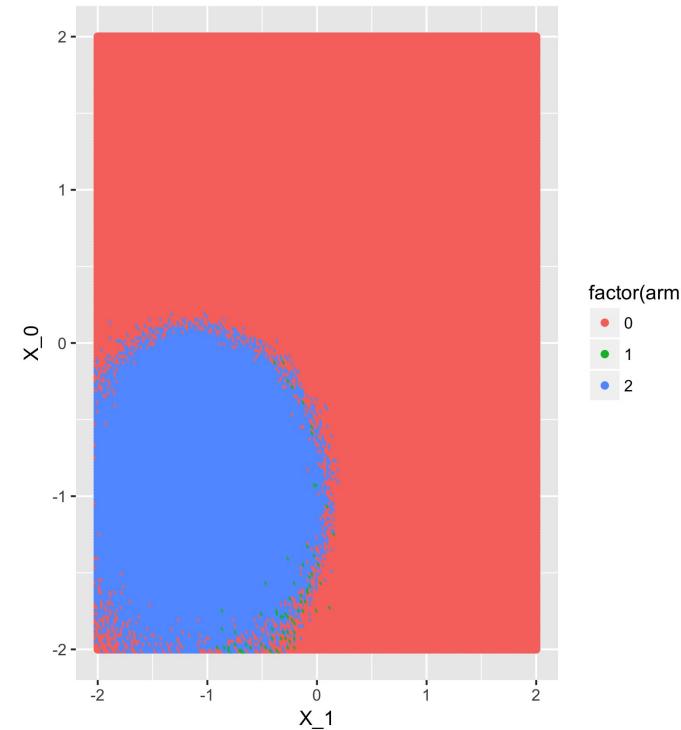


シミュレーション(左がUCB, 右がTS)

UCB Batch 13

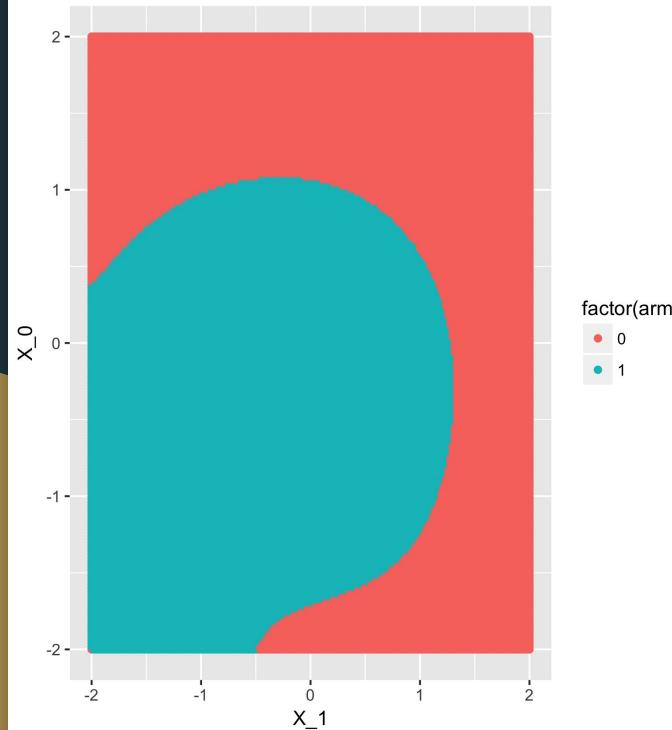


TS Batch 13

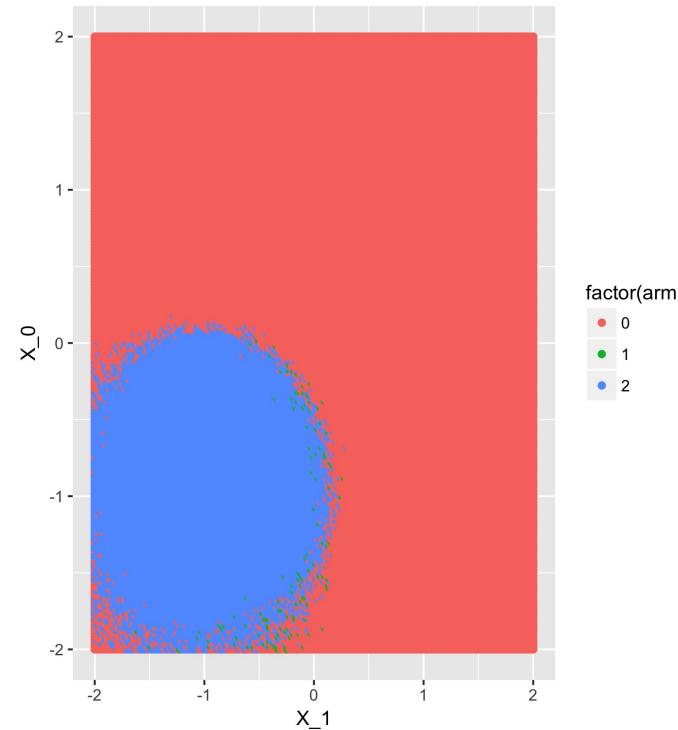


シミュレーション(左がUCB, 右がTS)

UCB Batch 14

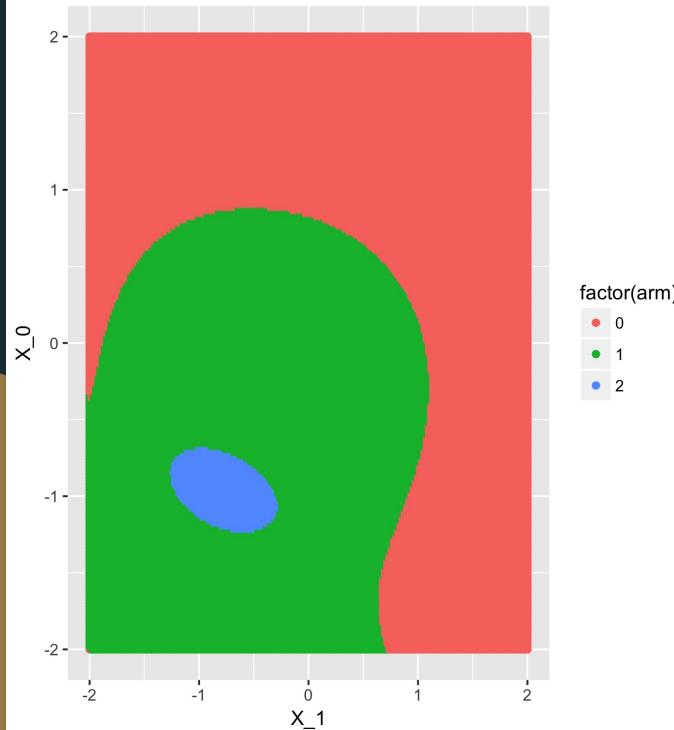


TS Batch 14

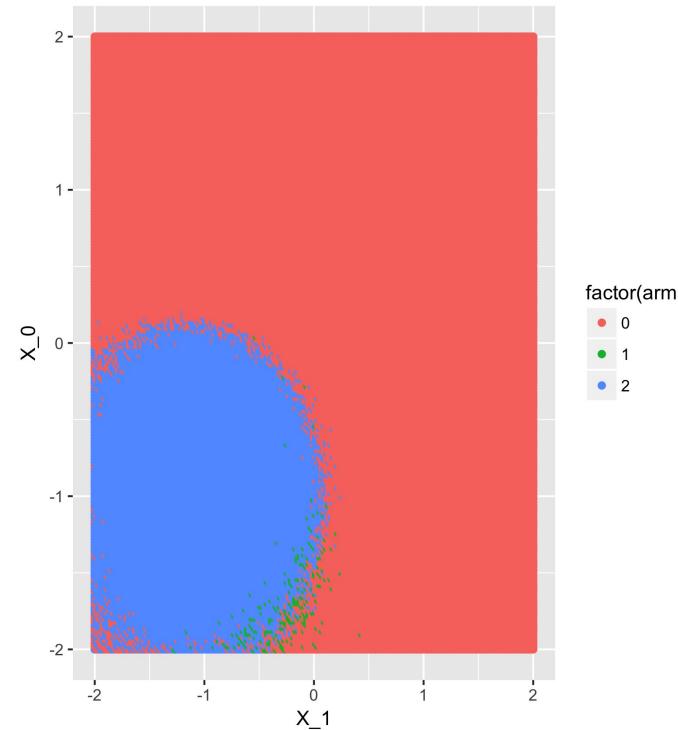


シミュレーション(左がUCB, 右がTS)

UCB Batch 15

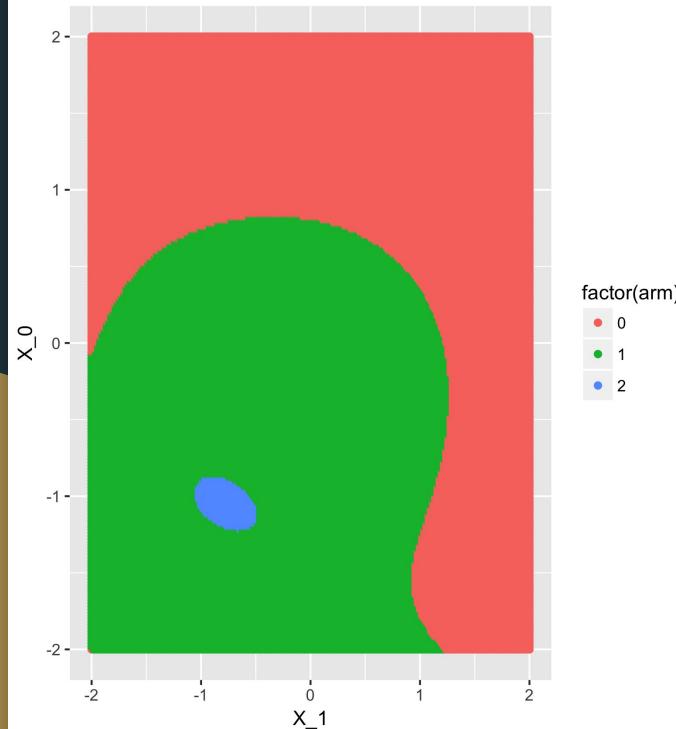


TS Batch 15

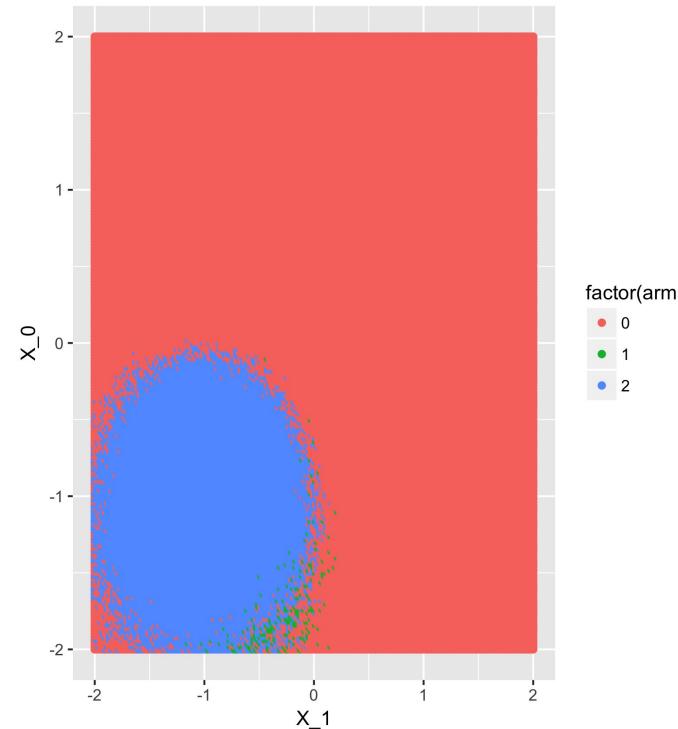


シミュレーション(左がUCB, 右がTS)

UCB Batch 16

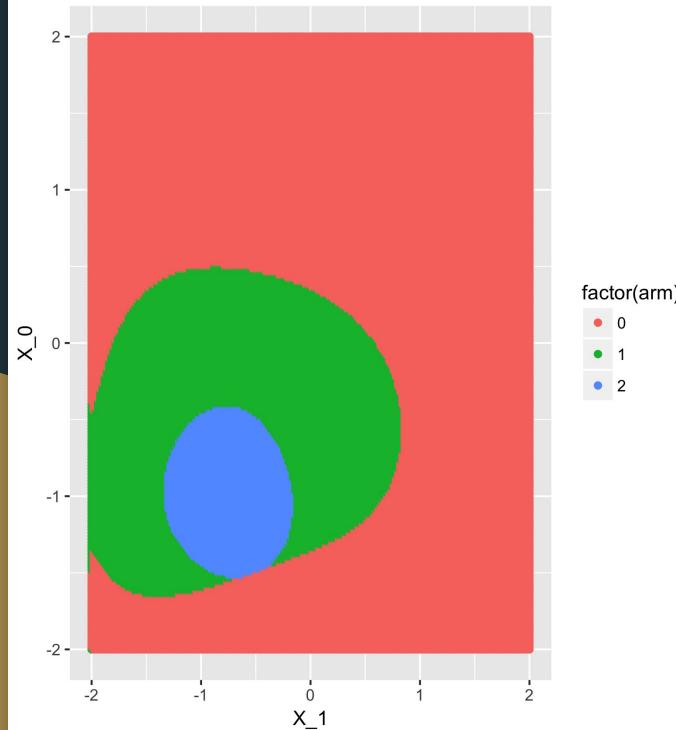


TS Batch 16

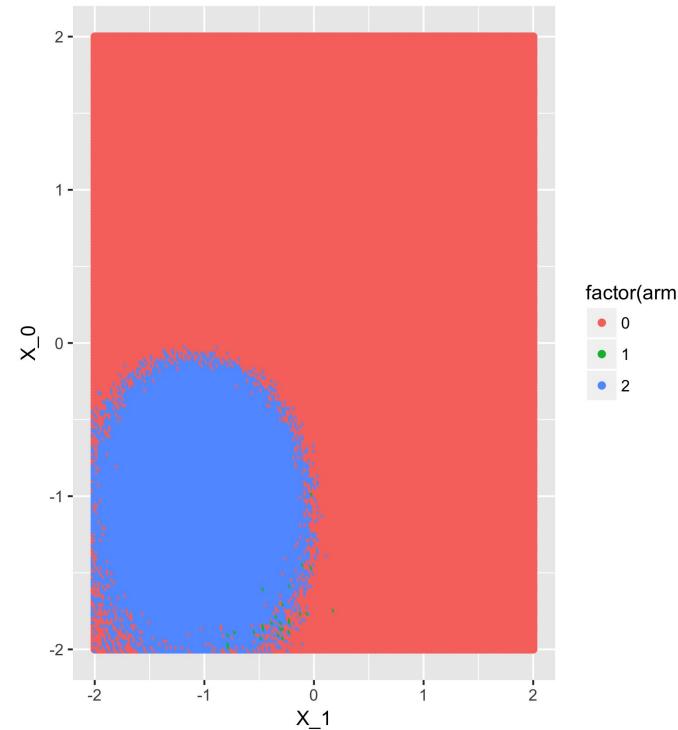


シミュレーション(左がUCB, 右がTS)

UCB Batch 17

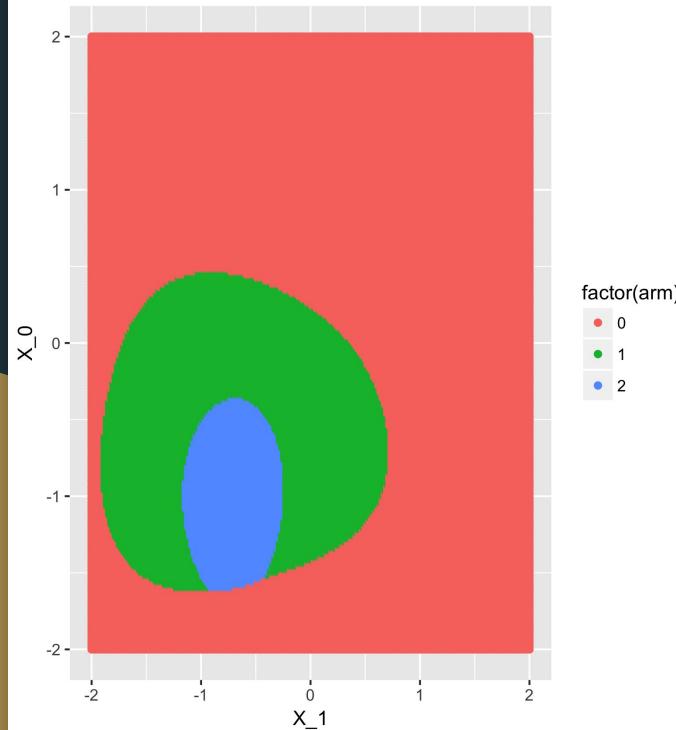


TS Batch 17

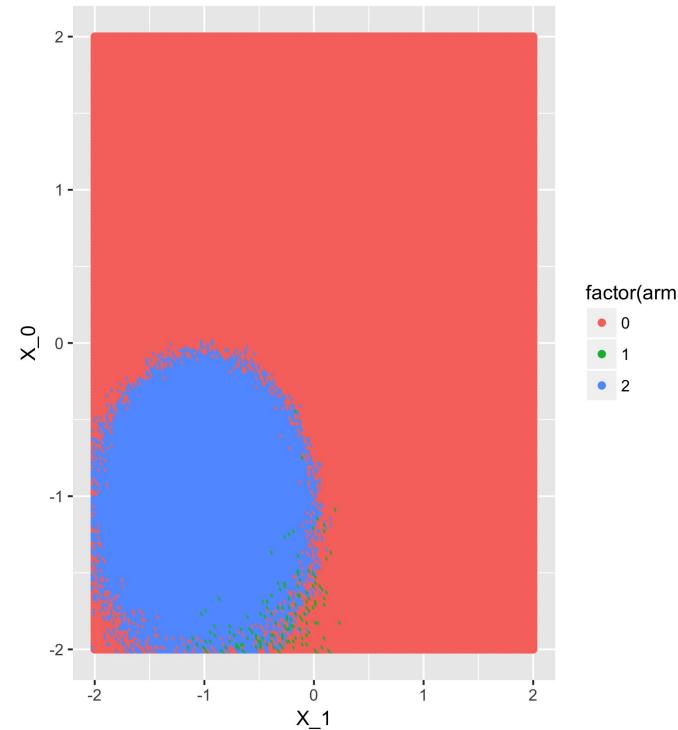


シミュレーション(左がUCB, 右がTS)

UCB Batch 18

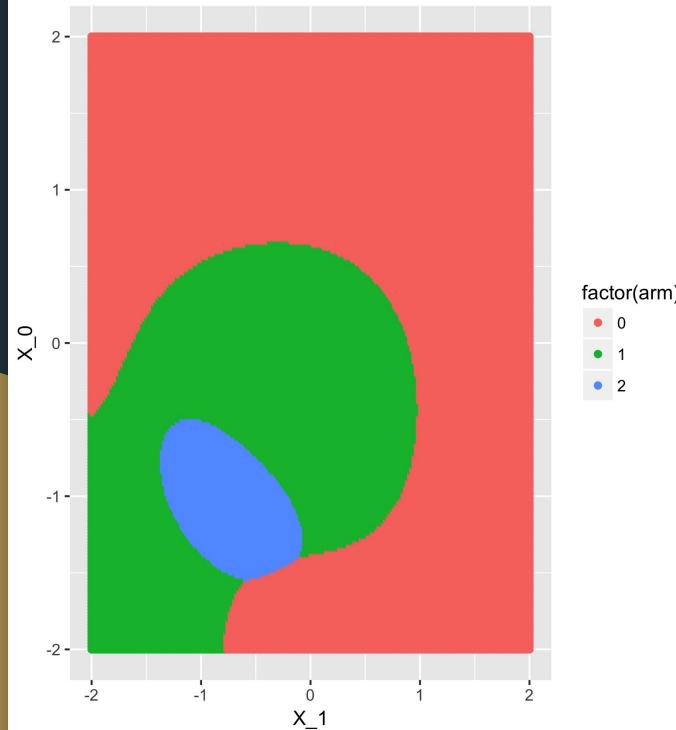


TS Batch 18

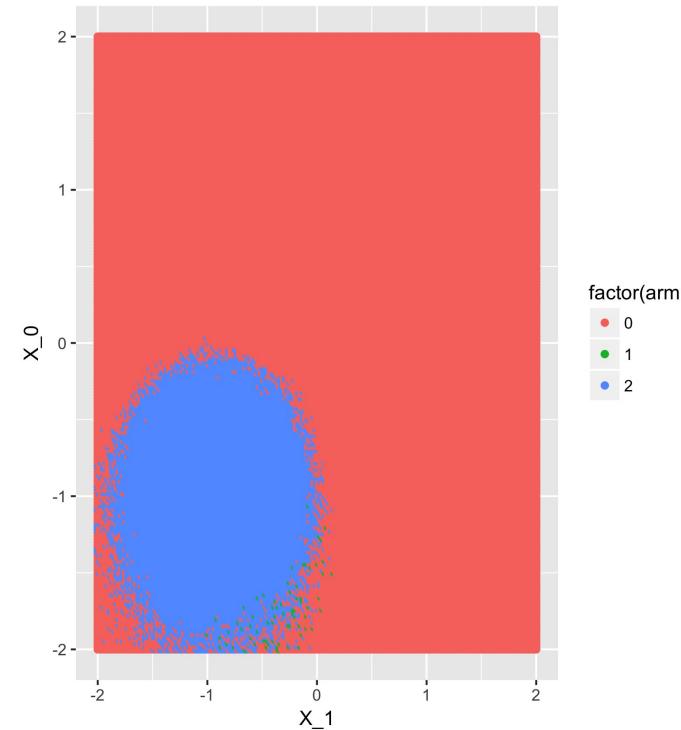


シミュレーション(左がUCB, 右がTS)

UCB Batch 19

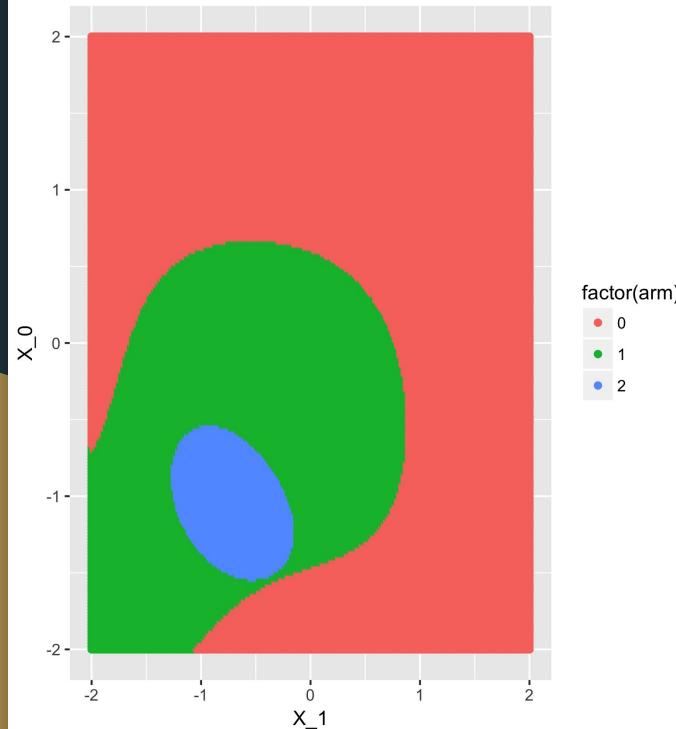


TS Batch 19

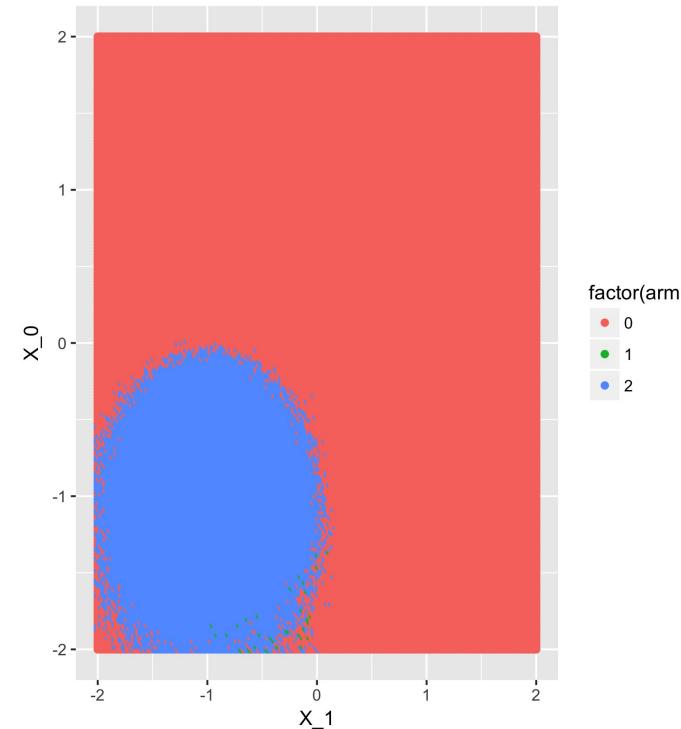


シミュレーション(左がUCB, 右がTS)

UCB Batch 20



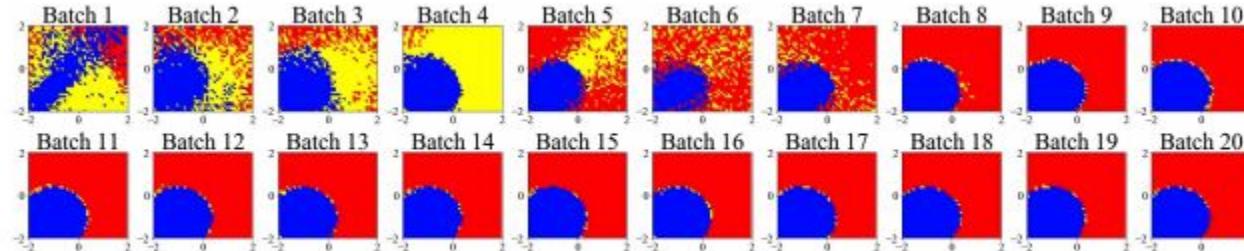
TS Batch 20



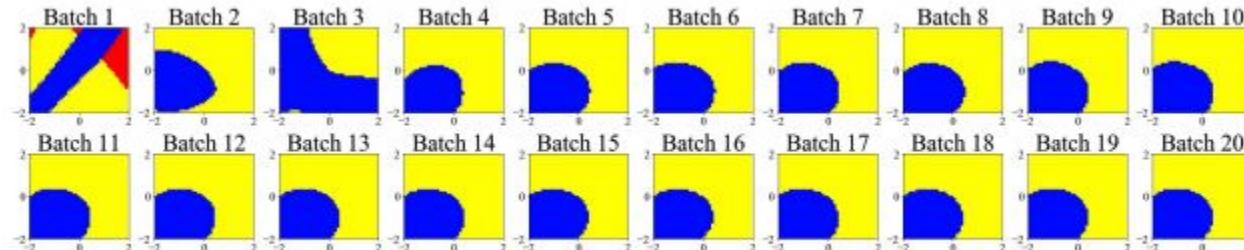
まとめ

- TSの方が、UCBよりよくなつた
- TSの方が良くなる理由としては、UCBだと決定的アルゴリズムなので、フィードバックループのような状況に陥りやすいという問題があるからと思われる
- (時間があれば)ノンパラの方の再現などもしてみたい

シミュレーション結果(論文)



(a) Well-Specified Ridge TS with Direct Model Estimation



(b) Well-Specified Ridge UCB with Direct Model Estimation

上がTS,下がUCB → TSの方がうまくいってる