離散構造の
オンライン予測

畑埜　晃平
九州大学

IBIS2013　企画セッション「学習理論」

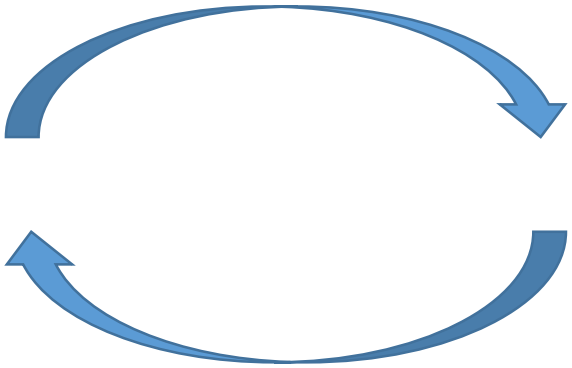
離散構造を用いた
“オンライン”意思決定問題

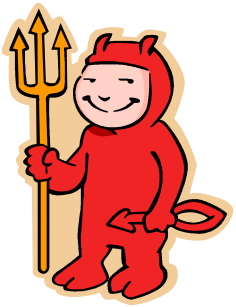
For t=1,…,T:

離散構造 $c_t \in C$

（敵対的）環境

プレイヤー





累積損失を
小さくしたい

損失 $l_t(c_t)$

具体例：資源割当，ネットワーク最適化，最短経路，
スケジューリング，ランキング

本発表：学習理論の立場からのサーベイ

目次

1. 離散構造のオンライン予測問題
2. 離散→連続ベクトル予測への帰着
3. オフライン－オンライン変換

例１：オンライン資源割当問題

For t=1,...,T日間:

1

バスを路線へ割当

プレイヤー（バス会社）

バス1

バス2

バス3

0.6
0.3
0.7
0.4
0.5
0.7
0.9
0.8
0.4

路線A

路線B

路線C

2

コストを割当

環境

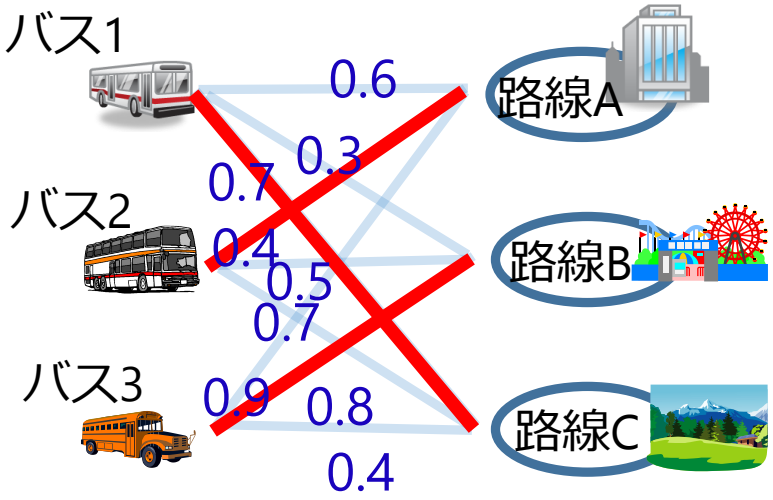
3 t日目におけるプレイヤーの損失=割当へのコストの和=0.7+0.4+0.8=1.9

例１：オンライン資源割当問題（定式化）

For t=1,…,T:

1 割当 ＝順列行列

$$\boldsymbol{c}_t = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$



2 コスト ＝コスト行列

$$\boldsymbol{l}_t = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.6 & 0.3 & 0.7 \\ 0.4 & 0.5 & 0.7 \\ 0.9 & 0.8 & 0.4 \end{pmatrix} \end{array}$$

3 t日目におけるプレイヤーの損失=

$$\boldsymbol{c}_t \bullet \boldsymbol{l}_t = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} 0.6 & 0.3 & 0.7 \\ 0.4 & 0.5 & 0.7 \\ 0.9 & 0.8 & 0.4 \end{pmatrix} = 0.7 + 0.4 + 0.8 = 1.9$$

決定空間 ＝順列行列全体

$$C = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \right\}$$

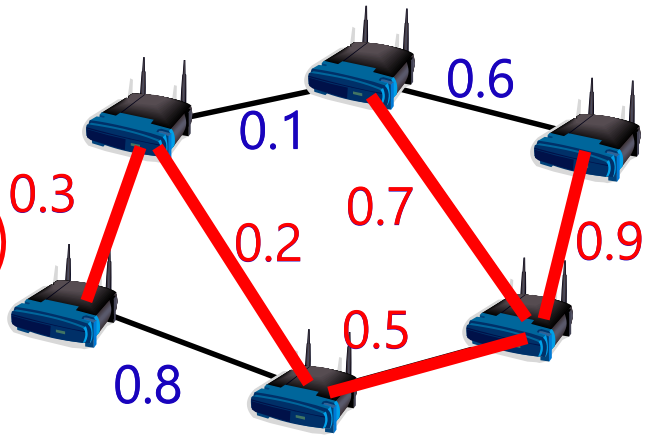
例２：オンラインルーティング問題

(spanning tree protocol)

For t=1,…,T日間:

1

ループのない部分ネットワーク
=全域木

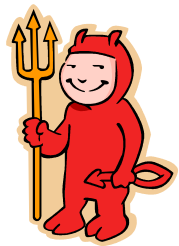


2

遅延を割当

プレイヤー
（プロトコル）

環境



3 t日目におけるプレイヤーの損失

=全域木中の遅延の総和=0.3+0.2+0.5+0.7+0.9=2.6

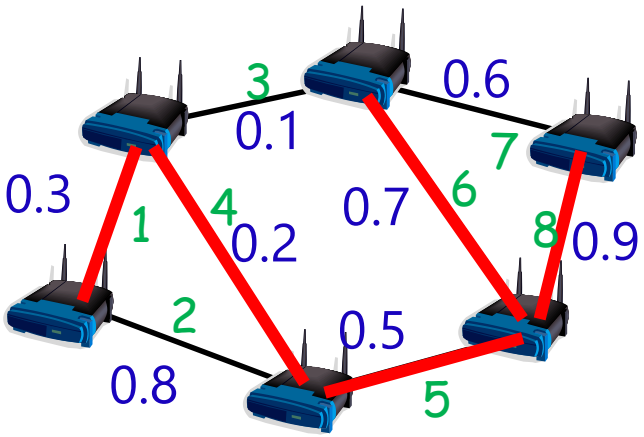
例２：オンラインルーティング問題（定式化）

For t=1,…,T日間:

1

全域木を表現するベクトル

$$\boldsymbol{c}_t = (1,0,0,1,1,1,0,1)$$
(1 2 3 4 5 6 7 8)



2

遅延を割当

$$\boldsymbol{l}_t = (0.3,0.8,0.1,0.2,0.5,0.7,0.6,0.9)$$
(1 2 3 4 5 6 7 8)

環境

プレイヤー（プロトコル）

3 t日目におけるプレイヤーの損失

$$\boldsymbol{c}_t \cdot \boldsymbol{l}_t = 0.3+0.2+0.5+0.7+0.9=2.6$$

決定空間
＝全域木を表現するベクトル全体

$$C = \{(1,0,0,1,1,1,0,1),\ldots\}$$

決定空間Cの具体例

- k-集合(要素数kの{1,...,n}の部分集合)

  [Warmuth, KuzuminJMLR08][Uchiya, Nakamura, Kudo ALT10]

  オンラインPCA, k-選択多腕バンディット

- 全域木

  [Cesa-Bianchi, Lugosi COLT09] [Koolen, Kivinen, Warmuth COLT10]　　ネットワーク最適化

- 順列行列・ベクトル

  [Helmbold, Warmuth JMLR09][Yasutake, H, Kijima, Takimoto, Takeda ISAAC11]

  [Yasutake, H, Takimoto, Takeda ACML12][Ailon arxiv13]

  資源割当，ランキング，スケジューリング

順列行列 $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$　　順列ベクトル $(1 \quad 3 \quad 2)$

- s-t パス

  [Takimoto, Warmuth COLT03] [Kalai & Vempala JCSS05]

  [Gyorgy, Linder, Lugosi, Ottucsak JMLR07][Koolen, Kivinen, Warmuth COLT10]

  最短経路

定式化：離散構造のオンライン予測問題

仮定: C $(\subset R_+^n)$　離散構造のクラス　（例：順列行列全体）

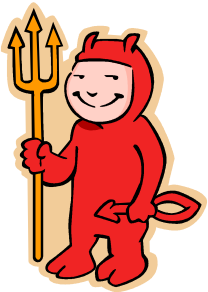
0　環境はT個の損失ベクトル $\boldsymbol{l}_t$ (t=1,...,T)を**前もって決める**

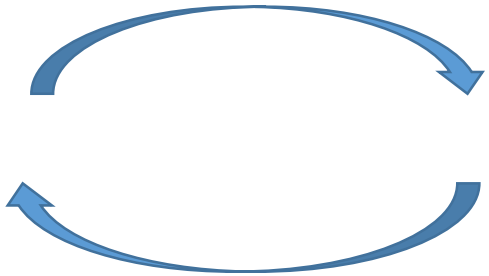
For t=1,...T

プレイヤー

1　離散構造　$\boldsymbol{c}_t \in C$

環境





2　損失ベクトル　$\boldsymbol{l}_t \in [0,1]^n$

3

プレイヤーの損失　$\boldsymbol{c}_t \cdot \boldsymbol{l}_t$

プレイヤーのゴール

$$\mathrm{Regret}(T) = E\left[\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{l}_t\right] - \min_{\boldsymbol{c} \in C} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{l}_t$$

リグレット

$$\mathrm{Regret}(T) = E\left[\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{l}_t\right] - \min_{\boldsymbol{c}\in C} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{l}_t$$

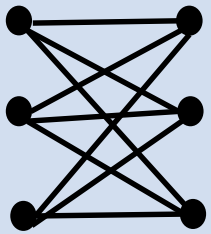
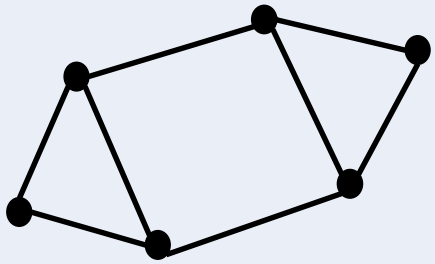
プレイヤー
（乱択アルゴリズム）の
期待累積損失

最適な固定の
離散構造の
累積損失

環境が前もって決めた（＝プレイヤーの動きは見ない）
最悪のT個の損失ベクトル $\boldsymbol{l}_t$ (t=1,…,T)に対する評価

さらなる仮定

- 決定空間Cは“非明示的”に与えられる

明示的に与えられる情報	決定空間C（非明示的）
完全２部グラフ 	順列行列全体 $C=\left\{\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix},\ldots,\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\right\}$ 資源割当
グラフ 	全域木を表現するベクトル全体 $C=\{(1,0,0,1,1,1,0,1),\ldots\}$ ルーティング

- 一般に|C|はnに関し指数的に大

ナイーブなアプローチ

- エキスパートを用いたオンライン予測手法
  - Hedge [Freund,Schapire JCSS97]
  - 各離散構造をエキスパートとみなして予測を統合
- リグレット：$O(\sqrt{T \log |C|})$ (ほぼ最適)
- １ステップあたりの計算時間$O(|C|)$

  ＝ 一般にnに関して指数時間
  - e.g. 資源割当:n!

問題の難しさとその対処

1. リグレットを小さくできるか？

Yes.
連続ベクトルのオンライン予測問題に帰着可能.
既存の連続ベクトル予測手法を用いて最適なリグレットを達成

2. 効率よく予測できるか？
   - 決定空間Cのサイズは指数的に大

場合によってはYes（本発表のフォーカス）

目次

1. 離散構造のオンライン予測問題
2. 離散→連続ベクトル予測への帰着
3. オフライン－オンライン変換

連続ベクトルのオンライン予測

Follow the Regularized Leader（FTRL）[Hazan 11]：

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{w \in W} \left(\sum_{j=1}^{t} \boldsymbol{w} \cdot \boldsymbol{l}_j + \eta D(\boldsymbol{w}, \boldsymbol{w}_0) \right)$$

累積損失

基準となるベクトルからの
Bregmanダイバージェンス

多くの場合，解析的な更新が可能

具体例：Hedge (KL-divergence) [Freund&Schapire JCSS97],
OGD (2-norm) [Zinkevich ICML03]

リグレット： $O(\sqrt{T})$ （最適）

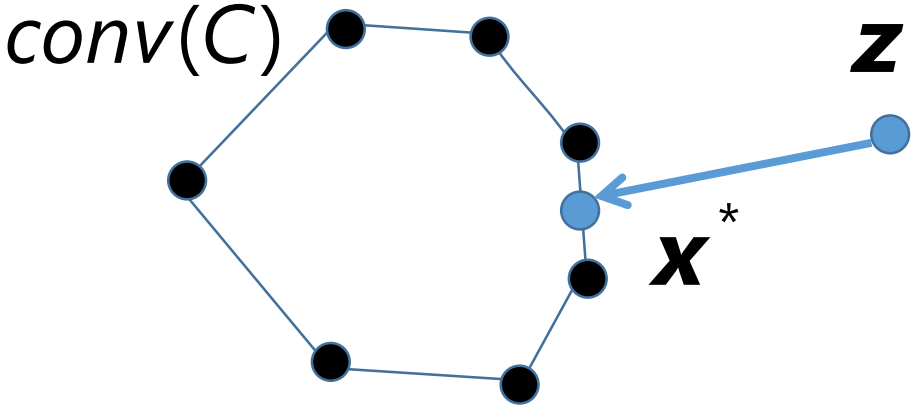
射影とラウンディング

conv(C): クラスC の要素の凸包

射影

入力： $\boldsymbol{z} \in R^n$

出力： $\boldsymbol{x}^* = \min_{x \in \mathrm{conv}(C)} D(\boldsymbol{x}, \boldsymbol{z})$



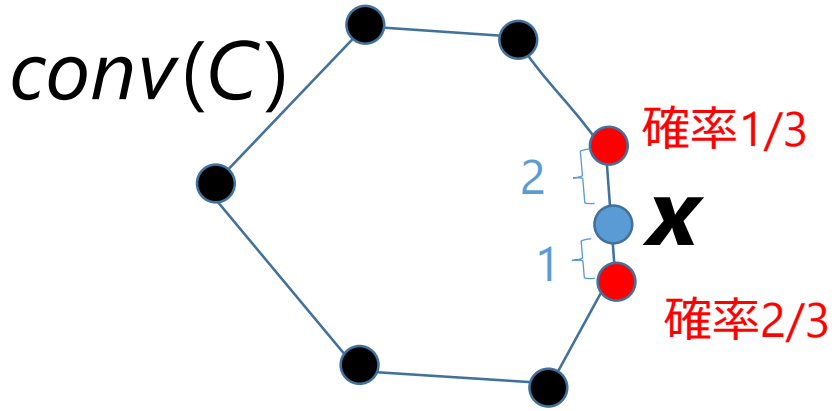
D : Bregmanダイバージェンス

(2 ノルム距離，KLダイバージェンス等)

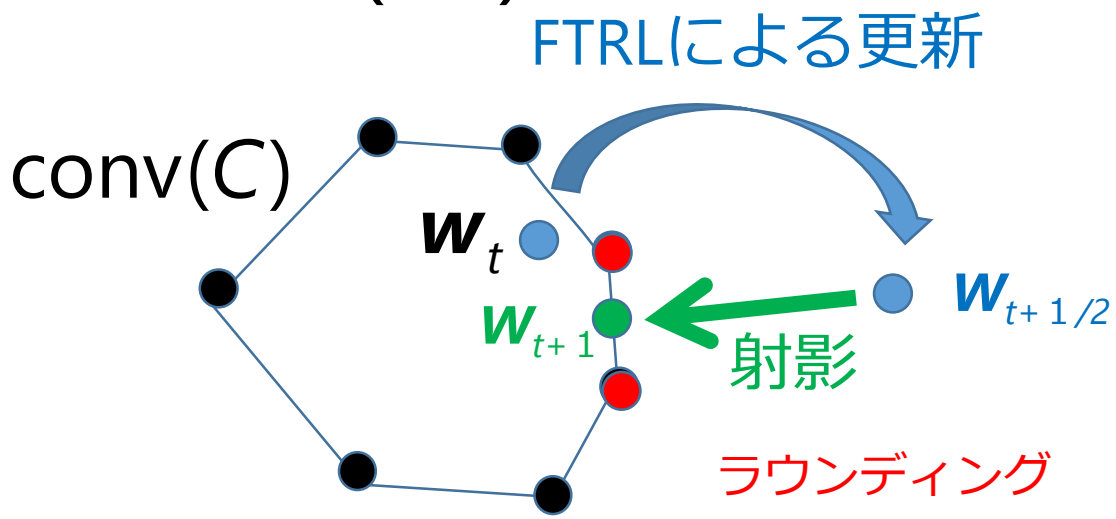
ラウンディング

入力： $\boldsymbol{x} \in \mathrm{conv}(C)$

出力： $\boldsymbol{c} \in C \ s.t. E(\boldsymbol{c}) = \boldsymbol{x}$



FTRL+射影＋ラウンディング
＝O(√T)リグレット



ラウンディングの性質
損失を保存

$$E(c_t \cdot l_t) = w_t \cdot l_t$$

離散→連続への帰着

離散構造の期待損失　conv(C)上のFTRLの損失

射影の性質
リグレットを保存

$$\begin{pmatrix}\text{conv(C)上の}\\ \text{FTRLのリグレット}\end{pmatrix} \leq \begin{pmatrix}\text{射影なしの}\\ \text{FTRLのリグレット}\end{pmatrix} = O(\sqrt{T})$$

C={順列行列}の場合(1)：射影

[Helmbold, Warmuth JMLR09]

- ２重確率行列：各列・行の総和が1となる正方正行列
- [Birkoffの定理] 任意の２重確率行列は順列行列の凸結合で表せる

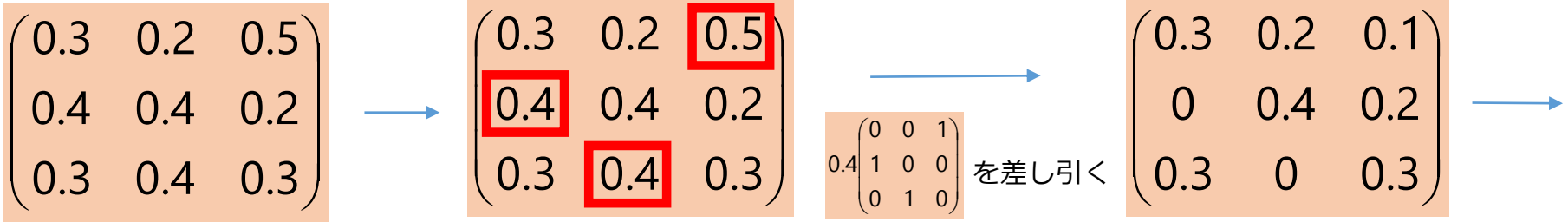
$$\begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0.4 & 0.4 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{pmatrix} = 0.4 \begin{pmatrix} & & 1 \\ 1 & & \\ & 1 & \end{pmatrix} + 0.3 \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} + 0.2 \begin{pmatrix} & 1 & \\ & & 1 \\ 1 & & \end{pmatrix} + 0.1 \begin{pmatrix} & & 1 \\ & 1 & \\ 1 & & \end{pmatrix}$$

- [系]conv(C)= ２重確率行列全体 (Birkoff polytope)
- conv(C)上の射影⇒ 2n個の線形制約つき凸最適化
- conv(C)へのKLダイバージェンス射影:
  Sinkhorn balancing, 計算時間 $\tilde{O}(n^6)$

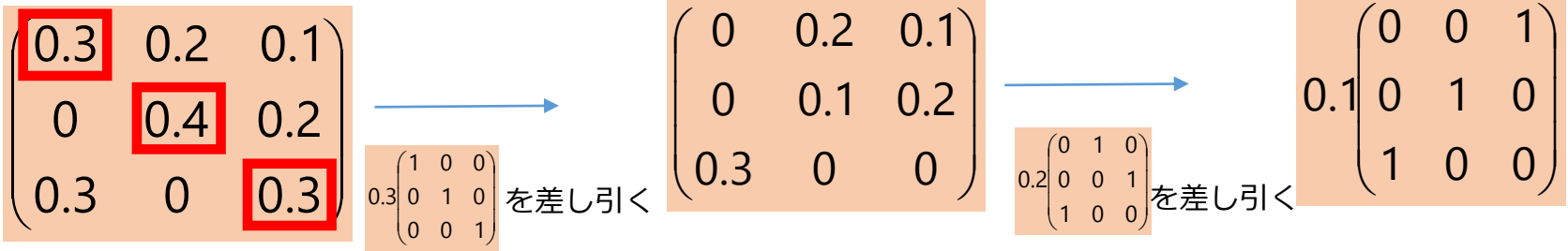
順列行列の場合(2)：ラウンディング

[Helmbold, Warmuth JMLR09]

- [Birkoffの定理，再掲] 任意の２重確率行列は順列行列の凸結合で表せる（存在性）
- 構成法（例）：



□は２部グラフ上の完全マッチングに対応
$O(n^3)$時間で求まる(Hangarian法など)



- １ステップ $O(n^3)$時間, n^2回の繰返し→ 全体$O(n^5)$時間

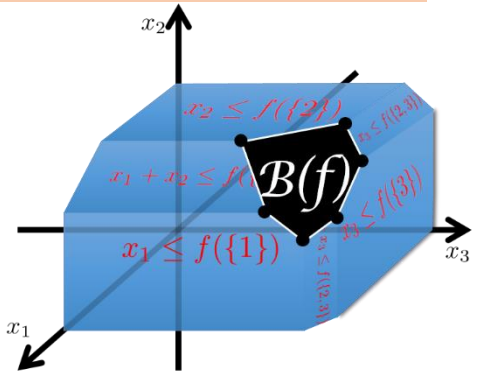
全域木・その他の場合

[Suehiro, H, Kijima,Takimoto, Nagano ALT12]

- 基多面体B(f)…劣モジュラ関数f:$2^{\{1,\ldots,n\}}$→Rによって定義される多面体:

$$B(f)=\left\{ \boldsymbol{x}\in R^n \mid \forall S\subseteq\{1,2,\ldots,n\}, \sum_{i\in S} x_i \le f(S), \sum_{i\in\{1,2,\ldots,n\}} x_i = f(\{1,2,\ldots,n\}) \right\}$$

conv(C)=B(f) となる決定空間Cの例：
全域木，k-set, 順列ベクトル



基多面体B(f)への射影・ラウンディング：O(poly(n))時間
特に f(S)=g(|S|)の場合（k-set, 順列ベクトル）：$O(n^2)$時間

ここまでのまとめ＆議論

射影・ラウンディングが計算できれば
離散構造に対する最適なリグレットを達成

<u>問題点</u>

- conv(C) の射影・ラウンディングがNP困難となる離散構造のクラスCが多数存在
  - オンライン集合被覆問題，オンラインMAXSATなど
- 離散構造ごとに射影・ラウンディングの設計が必要

<u>ではどうする？</u>

- 目標を弱める：α-リグレット
- “オフライン－オンライン変換”
  - 対応するオフライン手法があればオンライン手法に変換できる

目次

1. 離散構造のオンライン予測問題
2. 離散→連続ベクトル予測への帰着
3. オフライン－オンライン変換

オフライン近似アルゴリズムを用いた
離散構造のオンライン予測問題

オフライン問題

OPT(C)

入力：$\boldsymbol{l} \in [0,1]^n$

出力：$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{l}$

例：
集合被覆問題（配送計画）
MAXSAT（制約充足）
ランク統合問題（ランキング）

オンライン問題

プレイヤー

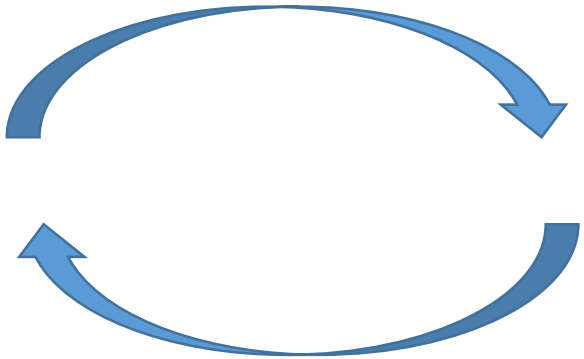
$\boldsymbol{l} \in [0,1]^n$

$\boldsymbol{c}$ s.t.

$\boldsymbol{c} \cdot \boldsymbol{l} \leq \alpha \arg\min_{\boldsymbol{c}' \in C} \boldsymbol{c}' \cdot \boldsymbol{l}$

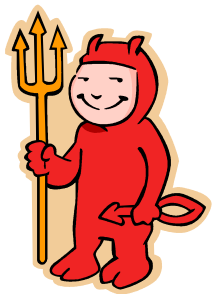
OPT（C)の
α近似
アルゴリズム

1 離散構造 $\boldsymbol{c}_t \in C$



2 損失ベクトル $\boldsymbol{l}_t \in [0,1]^n$

環境



3 プレイヤーの損失 $\boldsymbol{c}_t \cdot \boldsymbol{l}_t$

プレイヤーの目標

α-リグレットの最小化（なるべく小さい$\alpha \geq 1$に対して）

$$\alpha\text{-Regret}(T) = E\left[\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{l}_t\right] - \alpha \min_{\boldsymbol{c} \in C} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{l}_t$$

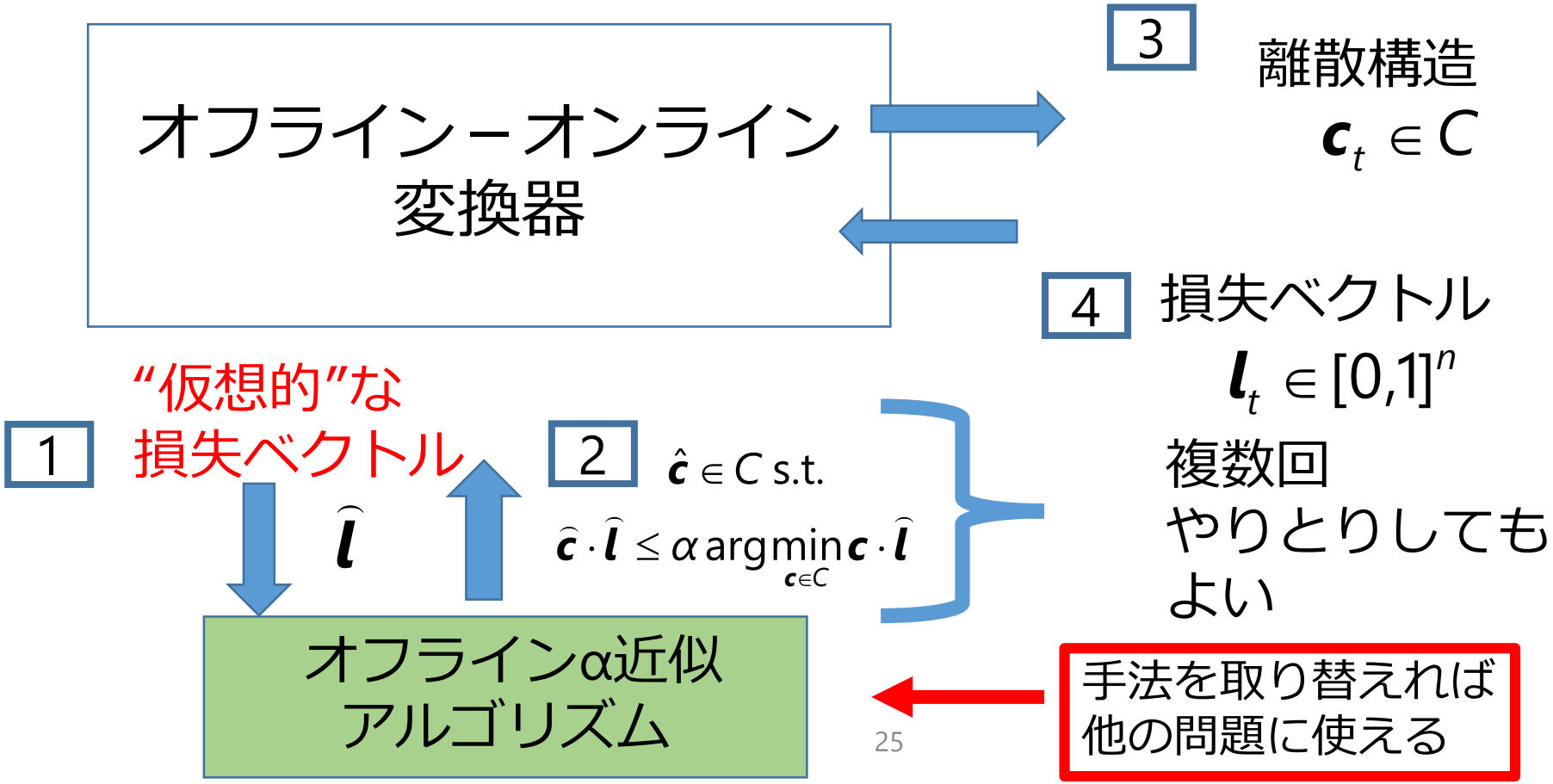
プレイヤーの
期待累積損失

最適な固定の
離散構造の
α近似による
累積損失

※リグレット＝1-リグレット

“オフライン－オンライン”変換

= オフライン近似アルゴリズムをオラクルとして用いた
オンライン予測手法



オフライン－オンライン変換手法

- Follow the Perturbed Leader (FPL) [Kalai,Vempala JCSS05]
  - オフライン最適化アルゴリズム（α=1）を用いる
  - 1-regret(T)=O($\sqrt{T}$) (ただしα-regret(T)=O($\alpha^T\sqrt{T}$)
  - 一般にFTRLより若干リグレットは悪化
  - 試行毎の計算時間: O(n)

- Kakadeらの手法 [Kakade, Kalai, Ligett SICOMP09]
  - オフラインα-近似アルゴリズムを用いる
  - α-regret(T)=O($\sqrt{T}$), 計算時間: poly(n)T

- Fujita-H-Takimoto[Fujita, H, Takimoto ALT13]
  - オフライン近似アルゴリズムが緩和に基づく（自然な仮定）
  - (α+ε)-regret(T)=O($\sqrt{T}$),計算時間: poly(n,1/ε) (Tに依存しない)

準備：Frank-Wolfe 法

[Frank,Wolfe56][Jaggi ICML13]

線形最適化手法を用いた凸最適化手法

$$\min_{\boldsymbol{x} \in P} f(\boldsymbol{x})$$

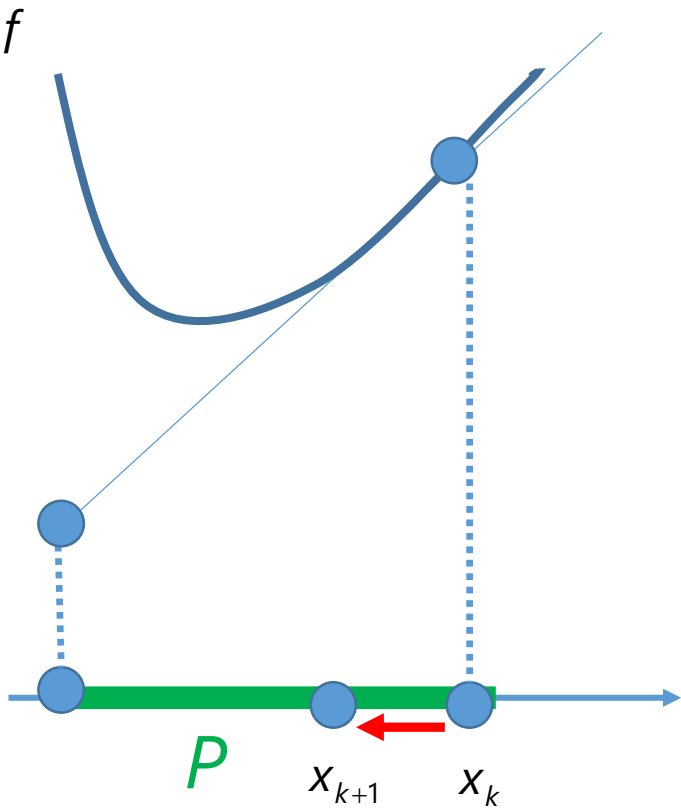
<u>Frank-Wolfe</u>

For k=1,….,K

a. $\gamma_k = 2/(k+2)$

b. $\boldsymbol{x} = \mathrm{argmin}_{\boldsymbol{x} \in P}\ \boldsymbol{x} \cdot \nabla f(\boldsymbol{x}_k)$

c. $\boldsymbol{x}_{k+1} = (1-\gamma_k)\boldsymbol{x}_k + \gamma_k \boldsymbol{x}$



<u>定理</u>

FW 法はK=O(1/ε) ステップで ε 近似解を出力

Kakade+：FW+近似アルゴリズム

[Kakade,Kalai,Ligett SICOMP09]

着眼点

$$\hat{\boldsymbol{c}} \cdot \hat{\boldsymbol{l}} \le \alpha \arg\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \hat{\boldsymbol{l}} = \arg\min_{\boldsymbol{c} \in \alpha C} \boldsymbol{c} \cdot \hat{\boldsymbol{l}}$$

近似解 $\hat{\boldsymbol{c}} \approx conv(\alpha C)$の端点

conv(αC）への射影を求める

$$\min_{x \in \alpha \mathrm{conv}(C)} f(x) = D(x, z)$$

For $k = 1,\ldots,K$

a. $\gamma_k = 2/(k+2)$

b. $\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c} \in \alpha C} \boldsymbol{c} \cdot \nabla f(\boldsymbol{x}_k)$ α近似アルゴリズムで解く

c. $\boldsymbol{x}_{k+1} = (1-\gamma_k)\boldsymbol{x}_k + \gamma_k \hat{\boldsymbol{c}}$ （実はラウンディング）

conv(αC）への射影・ラウンディングが同時に求まる

まとめ・今後の課題

まとめ

- 離散構造のオンライン予測
- FTRL+射影＋ラウンディング
- オフライン－オンライン変換

Take Home Message:　決定の候補が指数的に多くても
効率的にオンライン予測可能　・・・な場合もある

Open Problems

- 非線形な損失関数
- より効率的な手法
  - 緩和を仮定しない計算時間poly(n)のオフライン－オンライン変換は可能か？
- 確率的バンディット設定への拡張
- 応用