基礎講義

機械学習による発展的推定

Lab fOr
REsilient
& Future
City

第22回行動モデル夏の学校

2023/9/18

筑波大学　浦田淳司

今日の概要

1. パラメータ推定：深層学習と離散選択モデル

2. 先読み型の行動モデルのパラメータ推定：
動的離散選択モデルと強化学習

パラメータ推定とは

max（目的関数F）

パラメータβ

最終目標:

× パラメータを得る

○ モデル評価・予測を(高精度で)行う

パラメータ推定とは

max（目的関数F）
パラメータβ

連続量最適化

離散選択モデル – 尤度関数

$$\prod_i \delta_k^i P(k_i|\boldsymbol{x}_i, \boldsymbol{\beta})$$

機械学習・深層学習 – 損失関数

問題の性質により尤度関数や二乗誤差などを設定

$$\sum_i \|\boldsymbol{d}_i - y(\boldsymbol{x}_i|\boldsymbol{\beta})\|$$

i: サンプル、k: 選択肢、x: 属性、P: モデルによる選択確率、d: 目標出力量、y: モデルによる算出量

じゃぁ、なにが違うか？

離散選択モデル vs 機械学習・深層学習

モデル説明性（パラメータの意味、入力値影響）

明瞭

Black Box

→重要度スコア、SHAP(説明可能AI)

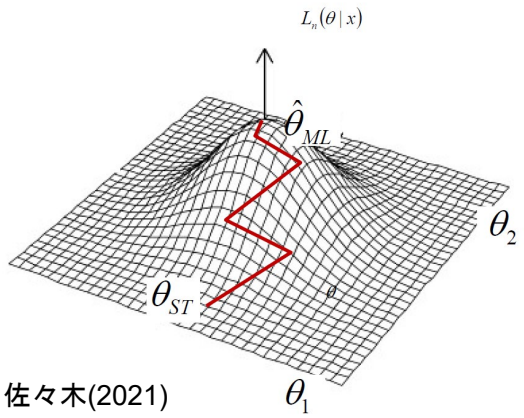
モデル予測性能

低い？

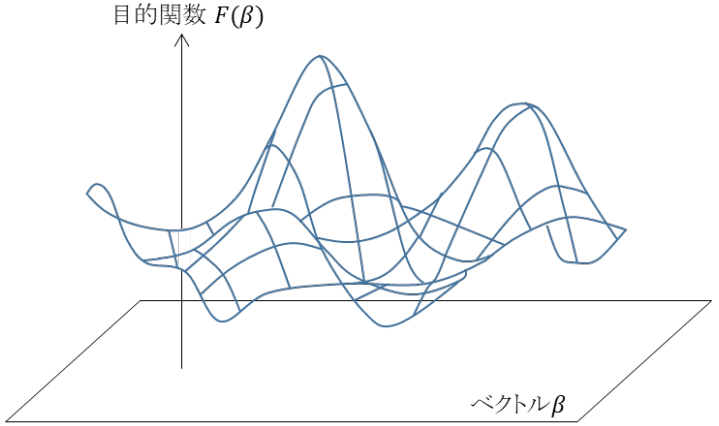
高い？

推定
アルゴリズム

モデル
アーキテクチャ



佐々木(2021)



た開発

っない

凸最適化⇔非凸最適化　モデル普遍性？

たくさんの最適化手法,,, [Schmidt et al. 2021]

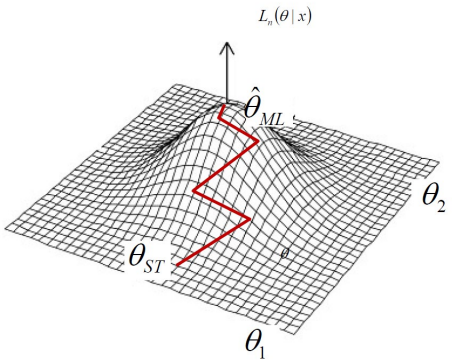
A. List of optimizers and schedules considered

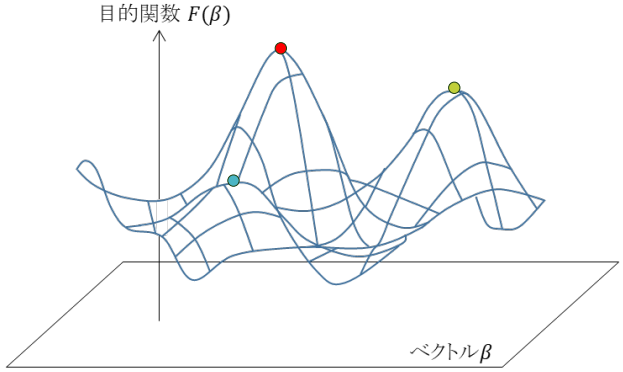
Table 2: List of optimizers considered for our benchmark. This is only a subset of all existing methods for deep learning.

Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KF-QN-CNN	(Ren & Goldfarb, 2021)
AdaBatch	(Devarakonda et al., 2017)	KFAC	(Martens & Grosse, 2015)
AdaBayes/AdaBayes-SS	(Aitchison, 2020)	KFLR/KFRA	(Botev et al., 2017)
AdaBelief	(Zhuang et al., 2020)	L4Adam/L4Momentum	(Rolínek & Martius, 2018)
AdaBlock	(Yun et al., 2019)	LAMB	(You et al., 2020)
AdaBound	(Luo et al., 2019)	LaProp	(Ziyin et al., 2020)
AdaComp	(Chen et al., 2018)	LARS	(You et al., 2017)
Adadelta	(Zeiler, 2012)	LHOPT	(Almeida et al., 2021)
Adafactor	(Shazeer & Stern, 2018)	LookAhead	(Zhang et al., 2019)
AdaFix	(Bae et al., 2019)	M-SVAG	(Balles & Hennig, 2018)
AdaFom	(Chen et al., 2019a)	MADGRAD	(Defazio & Jelassi, 2021)
AdaFTRL	(Orabona & Pál, 2015)	MAS	(Landro et al., 2020)
Adagrad	(Duchi et al., 2011)	MEKA	(Chen et al., 2020b)
ADAHESSIAN	(Yao et al., 2020)	MTAdam	(Malkiel & Wolf, 2020)
Adai	(Xie et al., 2020)	MVRC-1/MVRC-2	(Chen & Zhou, 2020)
AdaLoss	(Teixeira et al., 2019)	Nadam	(Dozat, 2016)
Adam	(Kingma & Ba, 2015)	NAMSB/NAMSG	(Chen et al., 2019b)
Adam$^+$	(Liu et al., 2020b)	ND-Adam	(Zhang et al., 2017a)
AdamAL	(Tao et al., 2019)	Nero	(Liu et al., 2021b)
AdaMax	(Kingma & Ba, 2015)	Nesterov	(Nesterov, 1983)
AdamBS	(Liu et al., 2020c)	Noisy Adam/Noisy K-FAC	(Zhang et al., 2018)
AdamNC	(Reddi et al., 2018)	NosAdam	(Huang et al., 2019)
AdaMod	(Ding et al., 2019)	Novograd	(Ginsburg et al., 2019)
AdamP/SGDP	(Heo et al., 2021)	NT-SGD	(Zhou et al., 2021b)
AdamT	(Zhou et al., 2020)	Padam	(Chen et al., 2020a)
AdamW	(Loshchilov & Hutter, 2019)	PAGE	(Li et al., 2020b)
AdamX	(Tran & Phong, 2019)	PAL	(Mutschler & Zell, 2020)
ADAS	(Eliyahu, 2020)	PolyAdam	(Orvieto et al., 2019)
AdaS	(Hosseini & Plataniotis, 2020)	Polyak	(Polyak, 1964)
AdaScale	(Johnson et al., 2020)	PowerSGD/PowerSGDM	(Vogels et al., 2019)
AdaSGD	(Wang & Wiens, 2020)	Probabilistic Polyak	(de Roos et al., 2021)
AdaShift	(Zhou et al., 2019)	ProbLS	(Mahsereci & Hennig, 2017)
AdaSqrt	(Hu et al., 2019)	PStorm	(Xu, 2020)
Adathm	(Sun et al., 2019)	QHAdam/QHM	(Ma & Yarats, 2019)
AdaX/AdaX-W	(Li et al., 2020a)	RAdam	(Liu et al., 2020a)

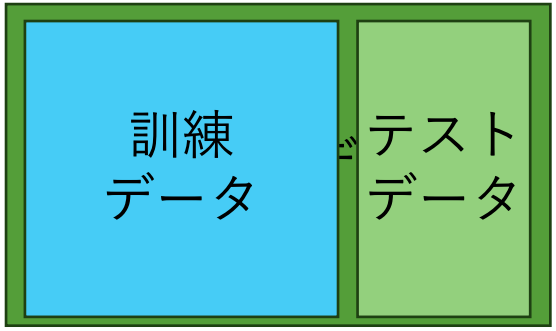
予測のための推定アルゴリズム開発①

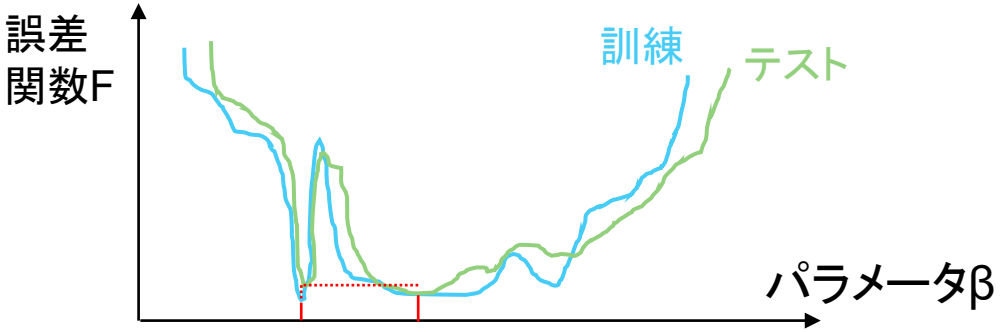
凸最適化⇔非凸最適化

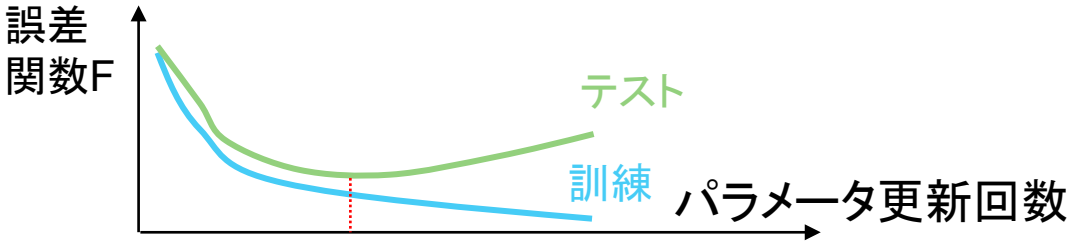




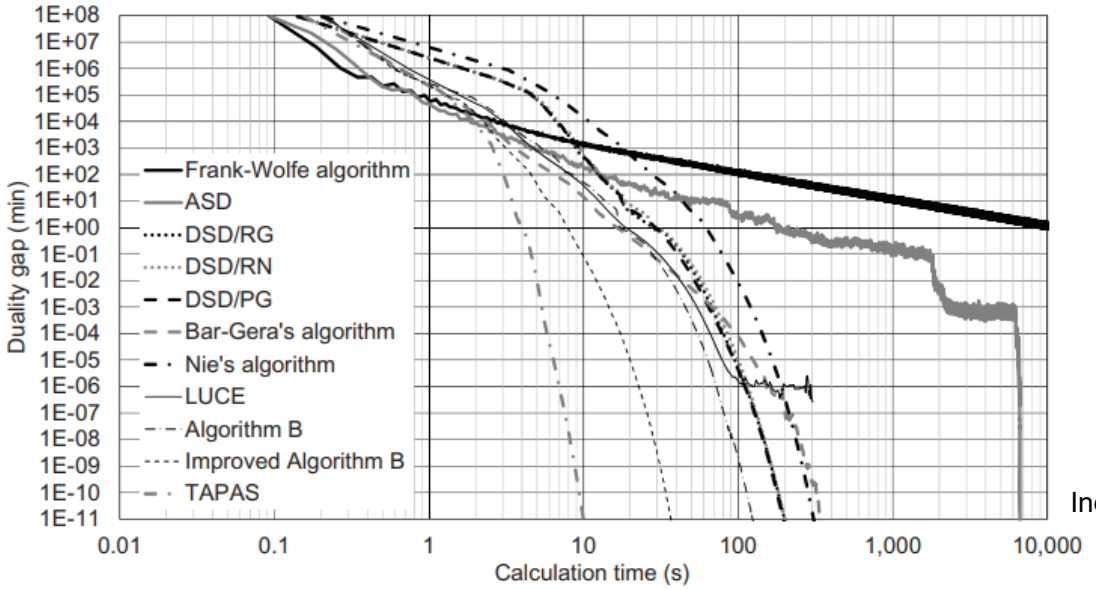
訓練誤差→テスト誤差→汎化誤差



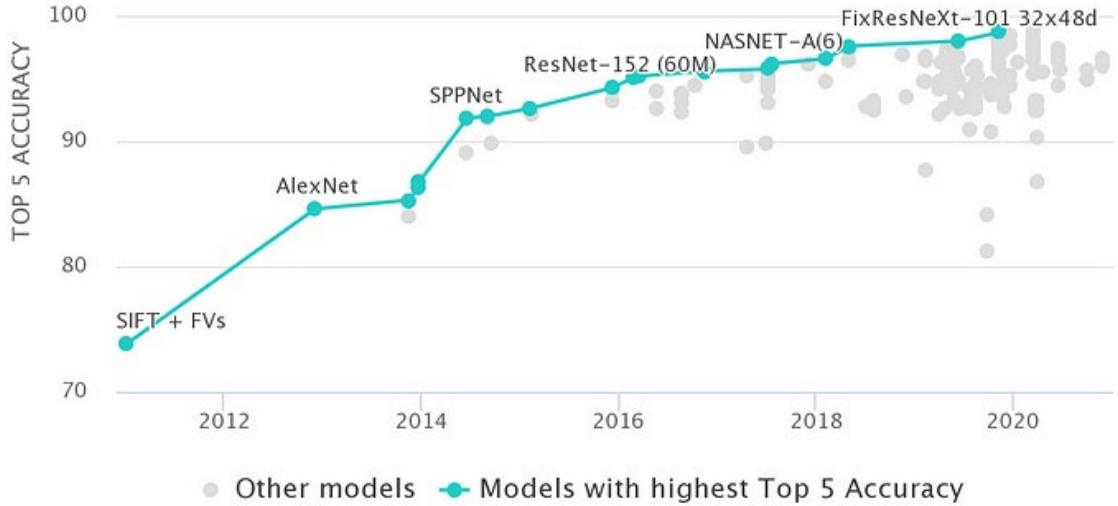




予測のための推定アルゴリズム開発①’



Inoue & Maruyama (2012)



https://towardsdatascience.com/accuracy-and-loss-things-to-know-about-the-top-1-and-top-5-accuracy-1d6beb8f6df3

データオープン
→競争的開発環境

誤差逆伝搬法①

大規模データを高速にパラメータ推定するための基本的なアイディア

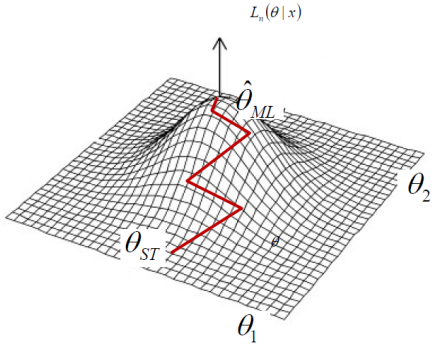
6

最大化アルゴリズムの考え方

佐々木(2021)

周りがあまり見えない中で，近傍の情報から頂点を目指す

- 対数尤度関数の段階的な最大化
  - 初期値を与える
  - 初期値周りで勾配(1次微分）等を用いて次の推定値の方向を決める
  - 初期値付近で１次微分，２次微分を用いて適切に次の点を決めて推定値を得る
  - 収束基準(一次微分ベクトル)で判定し，収束していない場合は，現在の値から次の推定値に移る

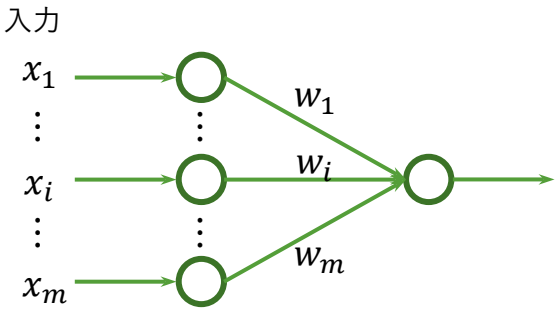


最適化計算のアルゴリズム：

- 繰り返し探索
  - 探索方向Δβの設定＋探索先での評価F
- 望ましい探索方向：最適解に近づく方向（≒微分方向）

誤差逆伝搬法②

ニューラルネットワーク



$$y = f(x = w_1x_1 + \cdots + w_mx_m - \theta) = \begin{cases} 1\ (x \geq 0) \\ 0 (x < 0) \end{cases}$$

一般化

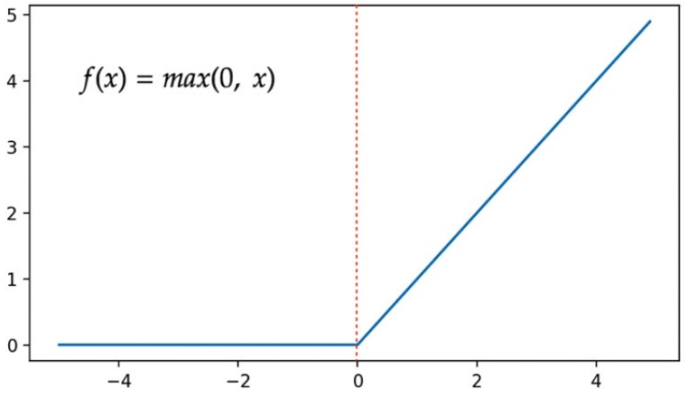
観測値 $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ 重み $\boldsymbol{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix}$

$$y = f(\boldsymbol{w}^T\boldsymbol{x} + b)$$

ステップ関数　バイアス

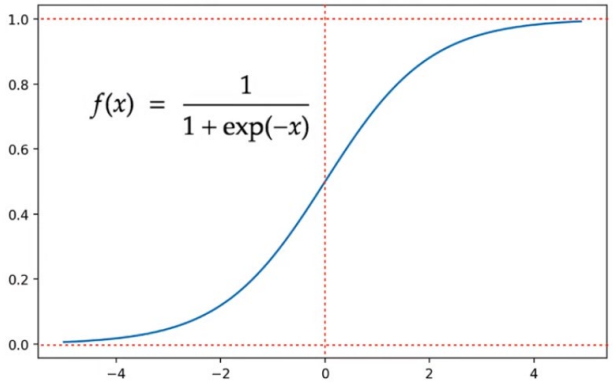
※ w, bがパラメータ

ステップ関数



https://onl.tw/r3E6iRE

シグモイド関数



$$\frac{\partial}{\partial x} f(x) = f(x)\big(1 - f(x)\big)$$

微分計算が楽

誤差逆伝搬法③ 降下勾配

損失関数 $F(\boldsymbol{w}, b) = \prod_i P(k_i|\boldsymbol{x}_i) = \prod_i y_i^{k_i}(1-y_i)^{1-k_i}$

対数損失関数 $E(\boldsymbol{w}, b) = -\log F(\boldsymbol{w}, b) = -\sum_n \{k_i \log y_i + (1-k_i)\log(1-y_i)\}$

反復計算によりパラメータを逐次的に更新（降下勾配法）

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \frac{\partial E(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} \qquad b^{t+1} = b^t - \eta \frac{\partial E(\boldsymbol{w}, b)}{\partial b}$$

η: 学習率（ハイパーパラメータ）※パラメータの更新幅

重みの勾配：シグモイド関数では，$\frac{\partial E(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} = \cdots = -\sum_n (k_i - y_i)\boldsymbol{x}_i$ となるため，

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \eta \sum_i (k_i - y_i)\boldsymbol{x}_i \qquad b^{t+1} = b^t + \eta \sum_i (k_i - y_i)$$

と式展開でき，$(k_i - y_i)$による差分と入力データ$\boldsymbol{x}_i$によりパラメータを更新

➡ 行列分解して並列計算できる ⇒ 簡単な並列計算に強みのあるGPUで高速化

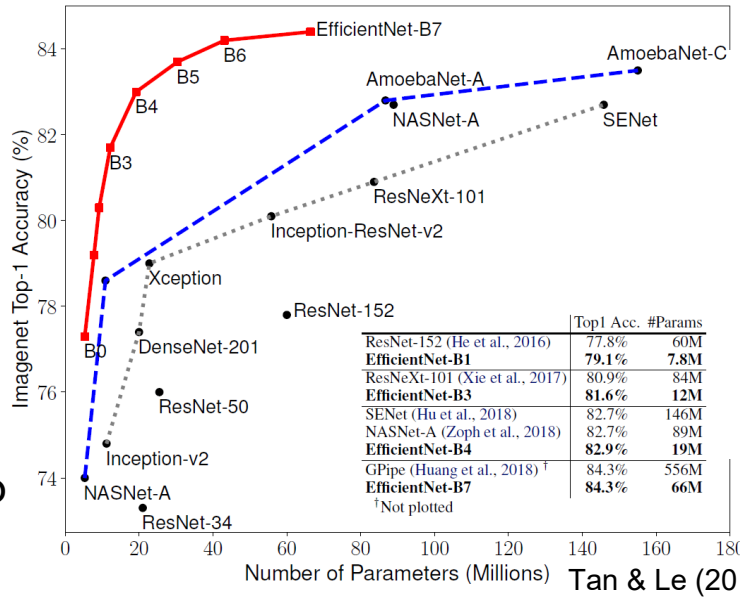
観測値 $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ 重み $\boldsymbol{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix}$

予測のための推定アルゴリズム開発②

■過剰パラメータ系での推定

詳細は今泉(2021)等

パラメータ数
≫
訓練サンプル数
⬇
無数の大域解
↓
正則モデル化による
信頼性向上へ

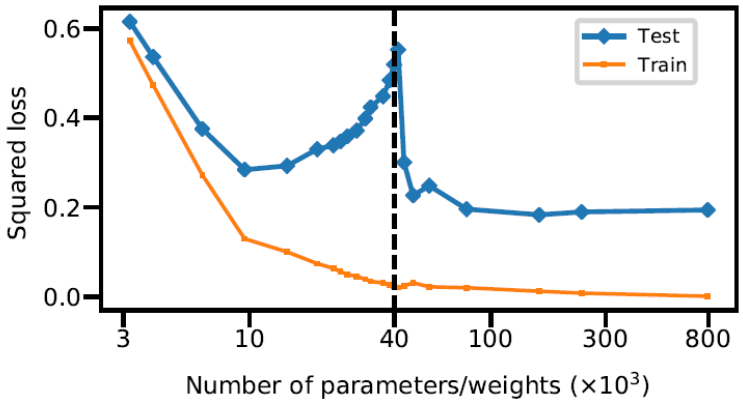


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

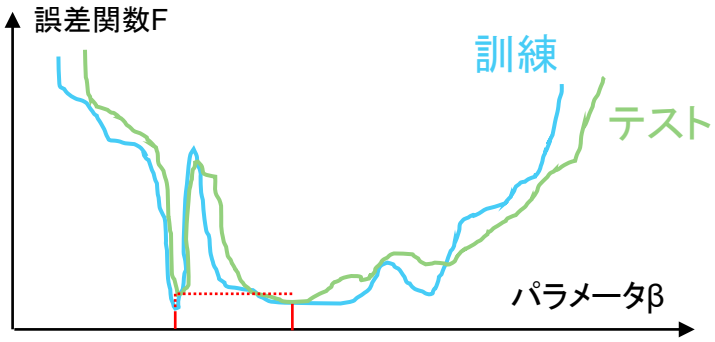
†Not plotted

Tan & Le (2019)

Double descent / 二重降下現象



■平坦性を評価：勾配のみに依存しない探索



・**Sharpness-aware Minimization（SAM）**
[Foret et al.(2020)]

フラットさの指標を**明示的に**最小化するアルゴリズム

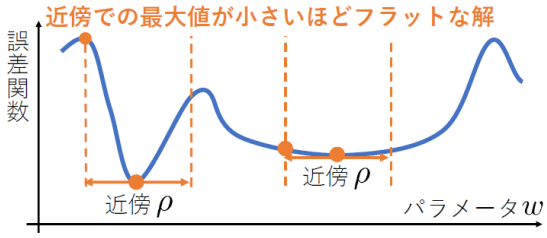
フラットさの指標

$$\max_{||\epsilon||\leq\rho} Loss(w+\epsilon)$$

（**近傍での誤差関数の最大値**）
が小さいほど解 w はフラットである

$$\theta_{t+1} = \theta_t - \eta\nabla_{Flat}L(\theta)$$

学習率　フラットさの指標の勾配ベクトル



今日の概要

1. パラメータ推定：深層学習と離散選択モデル

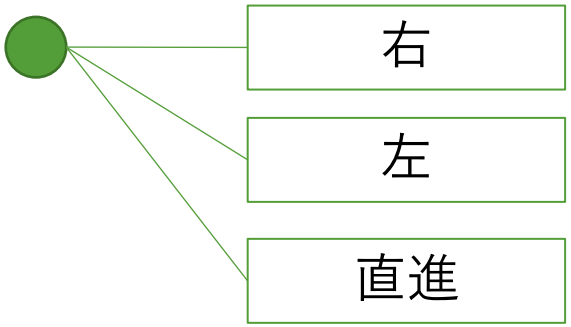
2. 先読み型の行動モデルのパラメータ推定：
動的離散選択モデルと強化学習

もちろん、推定アルゴリズムの開発は、
深層学習分野でしか行われていないわけではない

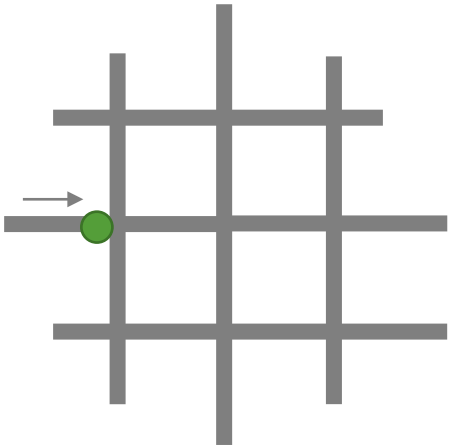
動学問題（動的選択問題）が難しいか。

静学問題

時刻t0

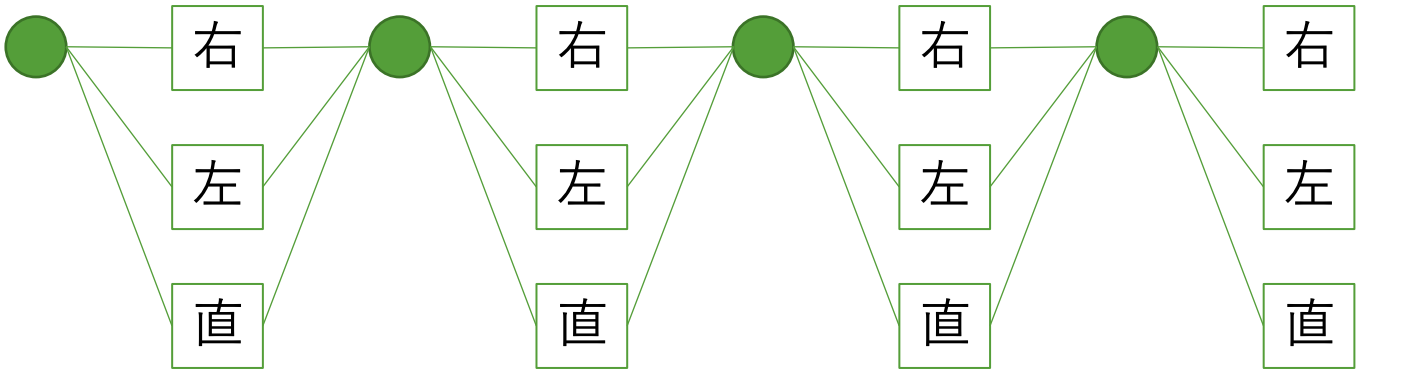


3選択肢



動学問題

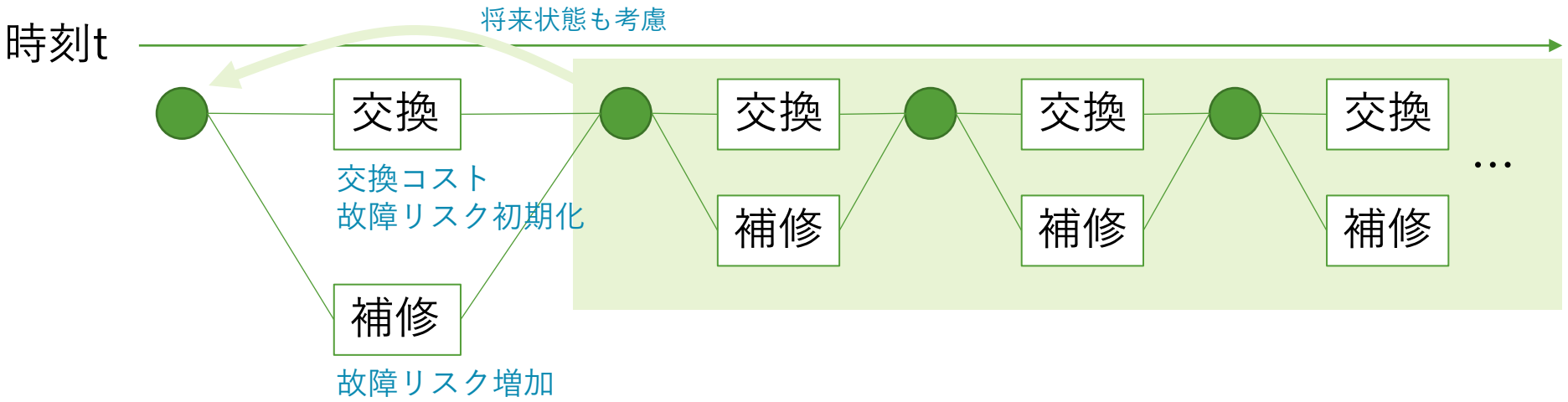
時刻t



3^4選択肢

動的離散選択モデル(概要)

Rust, J.(1987) のバスエンジン補修/交換の問題を例に説明



ある期の効用関数u

$$u(x, d, \varepsilon; \theta_1, \mathrm{RC}) = v(x, d; \theta_1, \mathrm{RC}) + \varepsilon(d)$$

$$v(x, d; \theta_1, \mathrm{RC}) = \begin{cases} -c(x; \theta_1), & \text{if } d = 0 \\ -\mathrm{RC} - c(0; \theta_1), & \text{if } d = 1 \end{cases}$$

状態xの推移確率(一次マルコフ性を仮定)

$$p(x_{t+1}, \varepsilon_{t+1} \mid x_t, \varepsilon_t, d_t; \theta_2, \theta_3)$$

d=1: 交換, d=0: 補修のみ
x: バスエンジンの状態
ε: 非観測項
c:補修コスト
θ_1: 補修パラメータ
RC: 交換コスト
θ_2, θ_3: 状態推移パラメータ

動的離散選択モデル(定式化)

将来の価値関数は，時間割引を考慮した効用の最大化となる

$$\mathrm{V}(x_t, \varepsilon_t) = \max_{\{d_t, d_{t+1}, d_{t+2}, \ldots\}} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} u(x_\tau, d_\tau, \varepsilon_\tau; \theta_1, \mathrm{RC})\right] \qquad (6)$$

時間割引率　$\beta \in (0,1)$

最適意思決定は無限期間先までを考慮しており，期によらず一定である

$$\mathrm{V}(x, \varepsilon) = \max_d \left\{ \underbrace{\nu(x, d; \theta_1, \mathrm{RC}) + \varepsilon(d)}_{\text{今期の効用}} + \beta \underbrace{\int_{x'} \int_{\epsilon'} \underbrace{\mathrm{V}(x', \varepsilon')}_{\text{次期の効用}} \underbrace{p(x', \varepsilon' | x, \varepsilon, d; \theta_2, \theta_3)}_{\text{今期の選択後の次期状態の推移確率}} dx'\, d\varepsilon'}_{\text{次期の期待価値関数}} \right\} \qquad (7)$$

次期状態(x', ε')

- 今期と次期の状態により，価値関数を定義
- 次期の期待価値関数の中に，次々期以降の効用は入っている

期待価値関数・選択確率

仮定：推移確率の条件つき独立性(CI)

$$p(x', \varepsilon' | x, \varepsilon, d; \theta_2, \theta_3) = p_2(\varepsilon' | x'; \theta_2) p_3(x' | x, d; \theta_3)$$

CIの仮定を用いて，期待価値関数を次で定義する．

$$\mathrm{EV}(x) = \int_{\varepsilon} \mathrm{V}(x, \varepsilon) p_2(\varepsilon | x; \theta_2) \, d\varepsilon$$

選択肢の期待価値関数（(7)式からεを除いて）

$$\mathrm{EV}(x, d) = \nu(x, d; \theta_1, \mathrm{RC}) + \varepsilon(d) + \beta \int_{x'} \mathrm{EV}(x') p_3(x' | x, d; \theta_3) \, dx'$$

これを期待価値関数の定義式に戻す

$$\mathrm{EV}(x) = \int_{\varepsilon'} \max_d \{\mathrm{EV}(x, d)\} p_2(\varepsilon' | x; \theta_2) \, d\varepsilon' \qquad (8)$$

εを極値分布とし，ロジット型の条件付き選択確率を導出（$\theta = (RC, \theta_1, \theta_3)$がパラメータ）

$$P(d | x; \theta) = \frac{\exp[\nu(x, d; \theta_1, \mathrm{RC}) + \beta \, \mathrm{EV}(x, d)]}{\sum_{d' \in \{0,1\}} \exp[\nu(x, d'; \theta_1, \mathrm{RC}) + \beta \, \mathrm{EV}(x, d')]} \qquad (9)$$

最尤法によるパラメータ推定

バス会社iの尤度　$$\ell_i(X^i;\theta)=\prod_{t=2}^{T}P(d_t^i|x_t^i;\theta)\,p_3(x_t^i|x_{t-1}^i,d_{t-1}^i;\theta_3)$$

M社の全体尤度　$$\ell(\theta)=\prod_{i=1}^{M}\ell_i(X^i;\theta)=\prod_{i=1}^{M}\prod_{t=2}^{T}P(d_t^i|x_t^i;\theta)\,p_3(x_t^i|x_{t-1}^i,d_{t-1}^i;\theta_3) \quad (14)$$

対数尤度　$$L(\theta)=\log\ell(\theta)$$

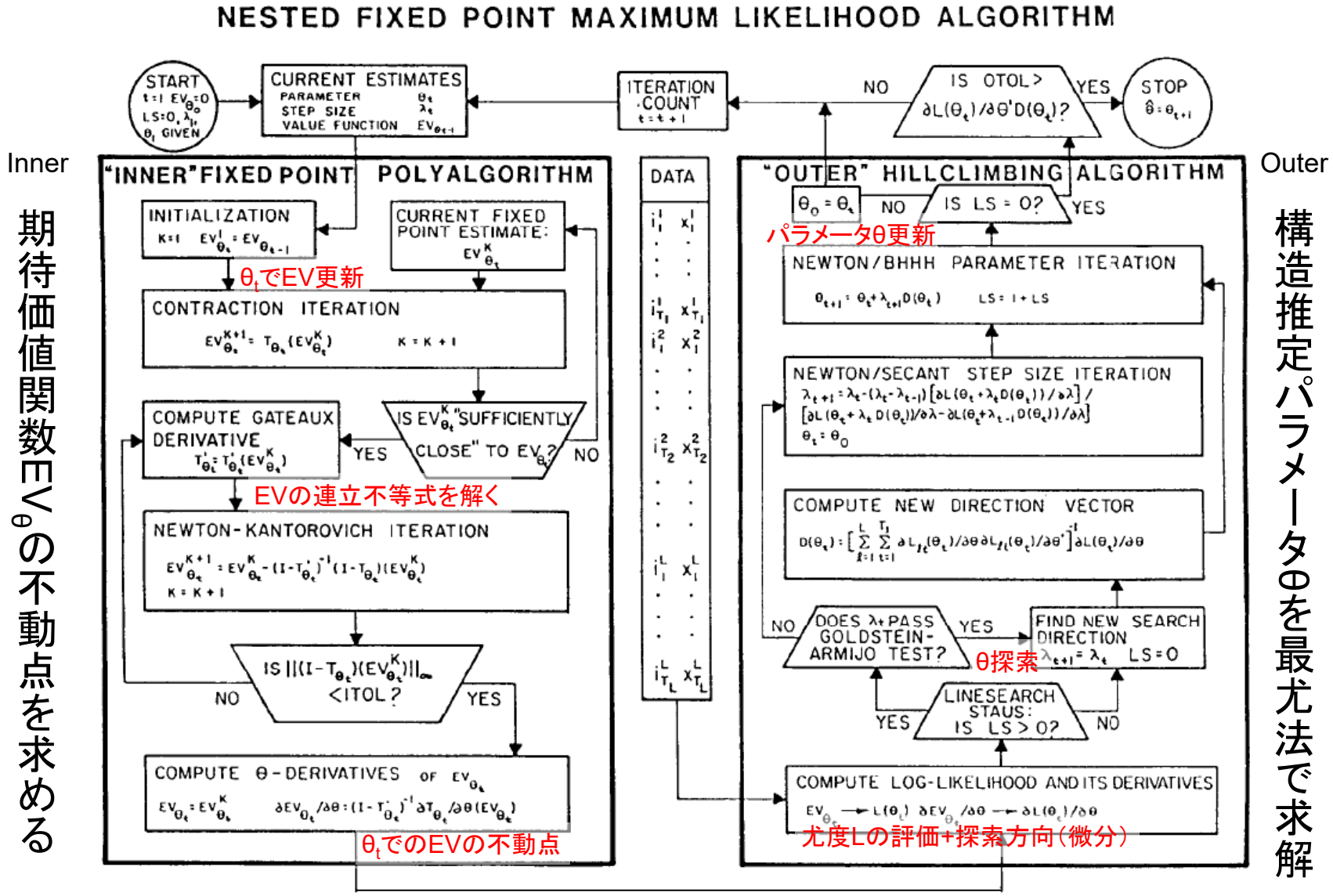
$$=\sum_{i=1}^{M}\sum_{t=2}^{T}\log[P(d_t^i|x_t^i;\theta)]+\sum_{i=1}^{M}\sum_{t=2}^{T}\log[p_3(x_t^i|x_{t-1}^i,d_{t-1}^i;\theta_3)] \quad (10)$$

対数尤度最大化により，$\theta=(RC,\theta_1,\theta_3)$を求める

$$\max_{\theta}\frac{1}{M}L(\theta) \quad (11)$$

推定方法１：NFXP法

Rust, J.(1988) Statistical Models of Discrete Choice Processes, Transportation Research Part B, Vol. 22(2), pp. 125-158.



NFXP法： Fixed-Point Solution

飛ばす？

(7)式と極値分布により，Fixed-Point equationは次となる

$$\mathrm{EV}(x,d)=\int_{x'=0}^{\infty}\log\left\{\sum_{d'\in\{0,1\}}\exp[\nu(x',d';\theta_1,\mathrm{RC})+\beta\,\mathrm{EV}(x',d')]\right\}\times p_3(dx'|x,d,\theta_3). \tag{12}$$

今期の状態xをK分割，次期の状態x'をJ分割して，表記.

$$p_3(x'|x_k,d,\theta_3)=\begin{cases}\Pr\{x'=\hat{x}_{k+j}|\theta_3\}, & \text{if } d=0\\ \Pr\{x'=\hat{x}_{1+j}|\theta_3\}, & \text{if } d=1\end{cases} \tag{13}$$

(11)式に代入

$$\mathrm{EV}(\hat{x}_k,d)=\sum_{j=0}^{J}\log\left\{\sum_{d'\in\{0,1\}}\exp[\nu(x',d';\theta_1,\mathrm{RC})+\beta\,\mathrm{EV}(x',d')]\right\}\times p_3(x'|\hat{x}_k,d,\theta_3). \tag{14}$$

※ この式が定義されている $\left[\mathrm{EV}=[\mathrm{EV}(\hat{x}_1,0),\ldots,\mathrm{EV}(\hat{x}_K,0),\mathrm{EV}(\hat{x}_1,1),\ldots,\mathrm{EV}(\hat{x}_K,1)]\right]$

式を簡略化

$$\mathrm{EV}=T(\mathrm{EV},\theta) \tag{15}$$

NFXP法：inner algorithm

飛ばす？

θ固定の下で，不動点 $EV_\theta = T_\theta(EV_\theta)$(15)' を求める
(15)は収縮写像であり，次が成立する.

$$||T_\theta(W) - T_\theta(V)|| \leq \beta||W - V|| \qquad (16)$$

この性質から，EV_kを次で更新する.

$$EV_{k+1} = T_\theta(EV_k) \qquad (17)$$

つまり，下記となる.

$$\begin{aligned} &|T_\theta(EV_{k+1}) - T_\theta(EV_k)| \leq \beta|EV_{k+1} - EV_k| \\ &\Leftrightarrow |EV_{k+2} - EV_{k+1}| \leq \beta|EV_{k+1} - EV_k| \end{aligned} \qquad (18)$$

βの速度でEV_θに近づく.
ただし，EV_θを得るためには，k=∞が必要であり，不動点近傍では収縮が遅くなる.
不動点近傍では，次式を用いる(Newton-Kantorovich法).

$$0 = (I - T_\theta)EV_{k+1} \sim (I - T_\theta)EV_k + (I - T'_\theta)(EV_{k+1} - EV_k) \qquad (19)$$

$$EV_{k+1} = EV_k - (I - T'_\theta)^{-1}(I - T_\theta)EV_k \qquad (20)$$

推定法2：NPL(擬似最尤推定法)

Aguirregabiria and Mira(2002)

Step1: EV_0をランダムに与える

Step2: EV_kを(9)式の右辺に代入し，尤度最大化するパラメータθ_kを求める

$$P(d|x;\theta) = \frac{\exp[\nu(x,d;\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x,d)]}{\sum_{d'\in\{0,1\}} \exp[\nu(x,d';\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x,d')]} \qquad (9)$$

Step3: EV_kとθ_kを用いて，(14)式よりEv_{k+1}を求める　⇔NFXPのInner Algorithm

$$\mathrm{EV}(\hat{x}_k,d) = \sum_{j=0}^{J} \log\left\{ \sum_{d'\in\{0,1\}} \exp[\nu(x',d';\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x',d')] \right\} \times p_3(x'|\hat{x}_k,d,\theta_3). \qquad (14)$$

収束判定: $|EV_{k+1} - EV_k|$と$|\theta_{k+1} - \theta_k|$が十分小さければ収束
収束していなければ，step2に戻る

無限回の繰り返し計算によって不動点を得ることができ，NFXPと漸近等価

NPL(擬似最尤推定法)とNFXPの比較

計算速度はパラメータが4つの場合は，NPLは9倍速い

TABLE I

MONTE CARLO EXPERIMENT

	Experiment design
Model:	Bus engine replacement model (Rust)
Parameters:	$\theta_0 = 10.47$; $\theta_1 = 0.58$; $\beta = 0.9999$
State space:	201 cells
Number observations:	1000
Number replications:	1000
Initial probabilities:	Kernel estimates

Monte Carlo distribution of MLE
(In parenthesis, percentages over the true value of the parameter)

	θ_0		θ_1
Mean Absolute Error:	2.08 (19.9%)		0.17 (29.0%)
Median Absolute Error:	1.56 (14.9%)		0.13 (22.7%)
Std. dev. estimator:	2.24 (21.4%)		0.16 (26.9%)
Policy iterations (avg.):		6.2	

Monte Carlo distribution of PI estimators (relative to MLE)
(All entries are 100* (K-PI statistic-MLE statistic)/MLE statistic)

		Estimators		
Parameter	Statistics	1-PI	2-PI	3-PI
θ_0	Mean AE	4.7%	1.6%	0.3%
	Median AE	14.2%	0.2%	−0.3%
	Std. dev.	6.8%	1.2%	0.3%
θ_1	Mean AE	18.7%	1.5%	0.2%
	Median AE	25.1%	0.7%	0.6%
	Std. dev.	11.0%	1.3%	0.2%

Nested Pseudo Likelihood Algorithm (NPL):

Let $\hat{\theta}_f$ be an estimate of θ_f. Start with an initial guess for the conditional choice probabilities, $P^0 \in [0,1]^{MJ}$. At iteration $K \geq 1$, apply the following steps:

Step 1: Obtain a new pseudo-likelihood estimate of α, α^K, as

$$\alpha^K = \arg\max_{\alpha \in \Theta} \sum_{i=1}^{n} \ln \Psi_{(\alpha,\hat{\theta}_f)}(P^{K-1})(a_i|x_i) \tag{11}$$

where $\Psi_\theta(P)(a|x)$ is the element (a, x) of $\Psi_\theta(P)$.

Step 2: Update P using the 'arg max' from step 2, i.e.

$$P^K = \Psi_{(\alpha^K,\hat{\theta}_f)}(P^{K-1}). \tag{12}$$

Iterate in K until convergence in P (and α) is reached.

推定法3：MPEC型アルゴリズム (Su & Judd (2012))

MPEC（Mathematical Programming with Equilibrium Constraints)を用いた求解

(11)式，(15)式を等価な均衡制約条件付き最適化問題と定式化.

（等価であることの証明あり）

$$\max_{(\theta, \mathrm{EV})} \frac{1}{M}\mathcal{L}(\theta, \mathrm{EV}) \qquad (11)$$

$$\text{subject to} \quad \mathrm{EV} = T(\mathrm{EV}, \theta) \qquad (15)$$

NFXPが不動点の算出プロセスを毎回解いているのに比べて，
Bellmanの等式(15)を一度評価すればよいので，計算負荷は小さい.

計算精度の比較

(Su & Judd (2012))

			Parameters						
			RC	θ_{11}	θ_{30}	θ_{31}	θ_{32}	θ_{33}	
β	Implementation	True values:	**11.726**	**2.457**	**0.0937**	**0.4475**	**0.4459**	**0.0127**	MSE
0.975	MPEC/AMPL	Mean	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		Std. dev.	(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC/MATLAB	Mean	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		Std. dev.	(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP/MATLAB	Mean	12.213	2.606	0.0943	0.4473	0.4445	0.0127	3.123
		Std. dev.	(1.617)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
0.980	MPEC/AMPL	Mean	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		Std. dev.	(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC/MATLAB	Mean	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		Std. dev.	(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP/MATLAB	Mean	12.139	2.579	0.0943	0.4473	0.4455	0.0127	2.866
		Std. dev.	(1.571)	(0.459)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
0.985	MPEC/AMPL	Mean	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		Std. dev.	(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC/MATLAB	Mean	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		Std. dev.	(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP/MATLAB	Mean	12.021	2.544	0.0943	0.4473	0.4455	0.0127	2.136
		Std. dev.	(1.368)	(0.411)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
0.990	MPEC/AMPL	Mean	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		Std. dev.	(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC/MATLAB	Mean	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		Std. dev.	(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP/MATLAB	Mean	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		Std. dev.	(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
0.995	MPEC/AMPL	Mean	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		Std. dev.	(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC/MATLAB	Mean	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		Std. dev.	(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP/MATLAB	Mean	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		Std. dev.	(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–

[a]For each β, there are 250 replications. Standard deviations are reported in parentheses. MSE is calculated by summing over all structural parameters.

- 250回計算の平均をとり，パラメータの平均，分散を比較
- βが大きい場合は，NFXPと同じ結果であるなど，両者の精度の差はほぼない

計算時間の比較

(Su & Judd (2012))

・MPECのほうがNFXPよりも計算時間は短い(AMPLで180倍以上，MATLABで3倍以上)

β	Implementation	Runs Converged (out of 1250 runs)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Contraction Mapping Iter.
0.975	MPEC/AMPL	1240	0.13	12.8	17.6	–
	MPEC/MATLAB	1247	7.90	53.0	62.0	–
	NFXP	998	24.60	55.9	189.4	134,748
0.980	MPEC/AMPL	1236	0.15	14.5	21.8	–
	MPEC/MATLAB	1241	8.10	57.4	70.6	–
	NFXP	1000	27.90	55.0	183.8	162,505
0.985	MPEC/AMPL	1235	0.13	13.2	19.7	–
	MPEC/MATLAB	1250	7.50	55.0	62.3	–
	NFXP	952	43.20	61.7	227.3	265,827
0.990	MPEC/AMPL	1161	0.19	18.3	42.2	–
	MPEC/MATLAB	1248	7.50	56.5	65.8	–
	NFXP	935	70.10	66.9	253.8	452,347
0.995	MPEC/AMPL	965	0.14	13.4	21.3	–
	MPEC/MATLAB	1246	7.90	59.6	70.7	–
	NFXP	950	111.60	58.8	214.7	748,487

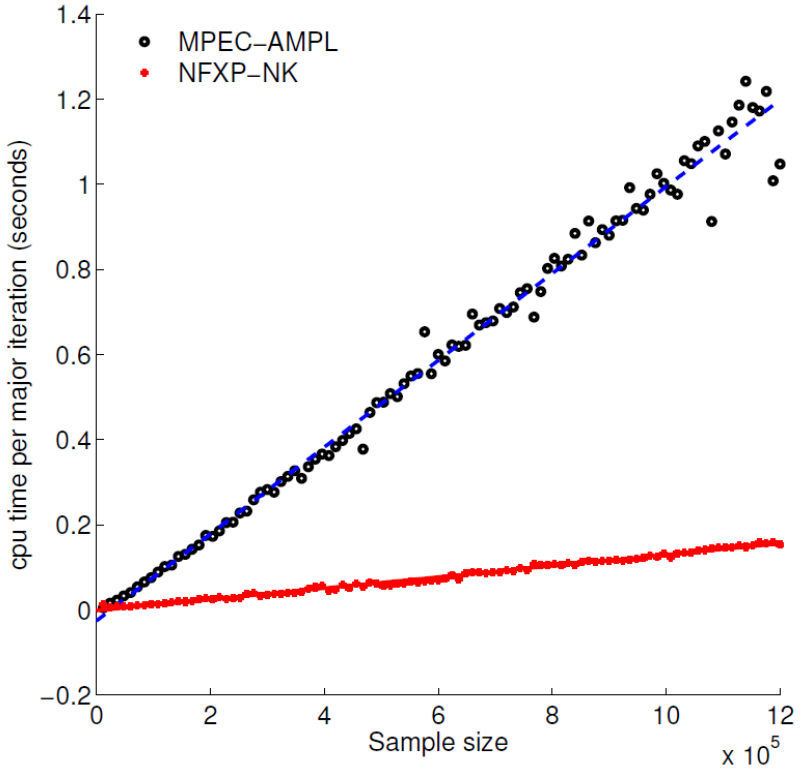
[a] For each β, we use five starting points for each of the 250 replications. CPU time, number of major iterations, number of function evaluations and number of contraction mapping iterations are the averages for each run.

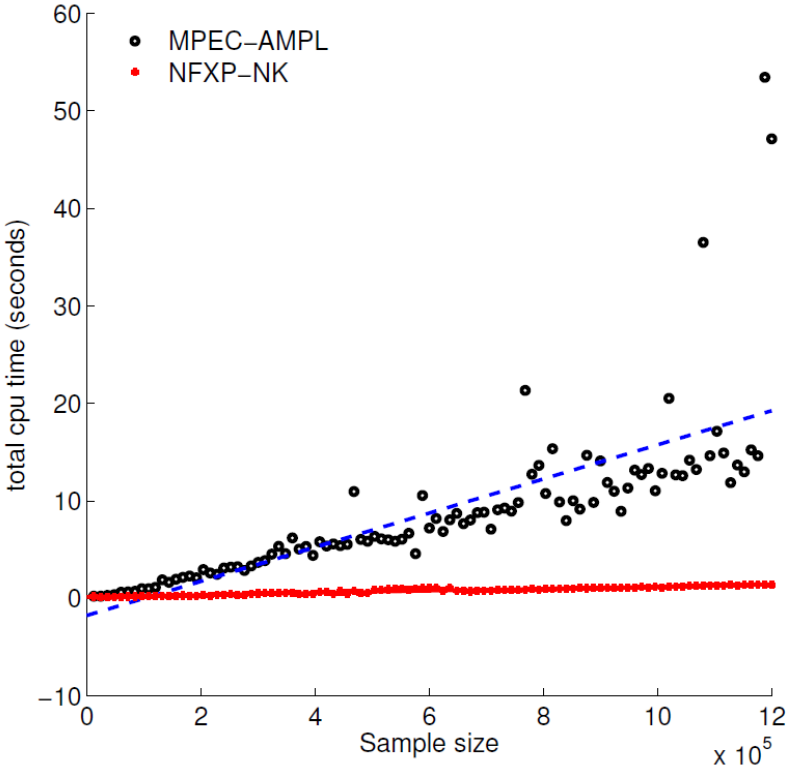
計算時間の比較（反論）

Iskhakov, Lee, Rust et al. (2016)

・MPECの比較対象とした推定アルゴリズムは古い
・MPECはサンプル数（状態数）＝等価制約数であり、サンプルが増えるほど、計算コスト増
・だけど、特殊な推定手法（NFXP）ではなく、一般的な最適化手法（MPEC）で解けるようにした点は長所

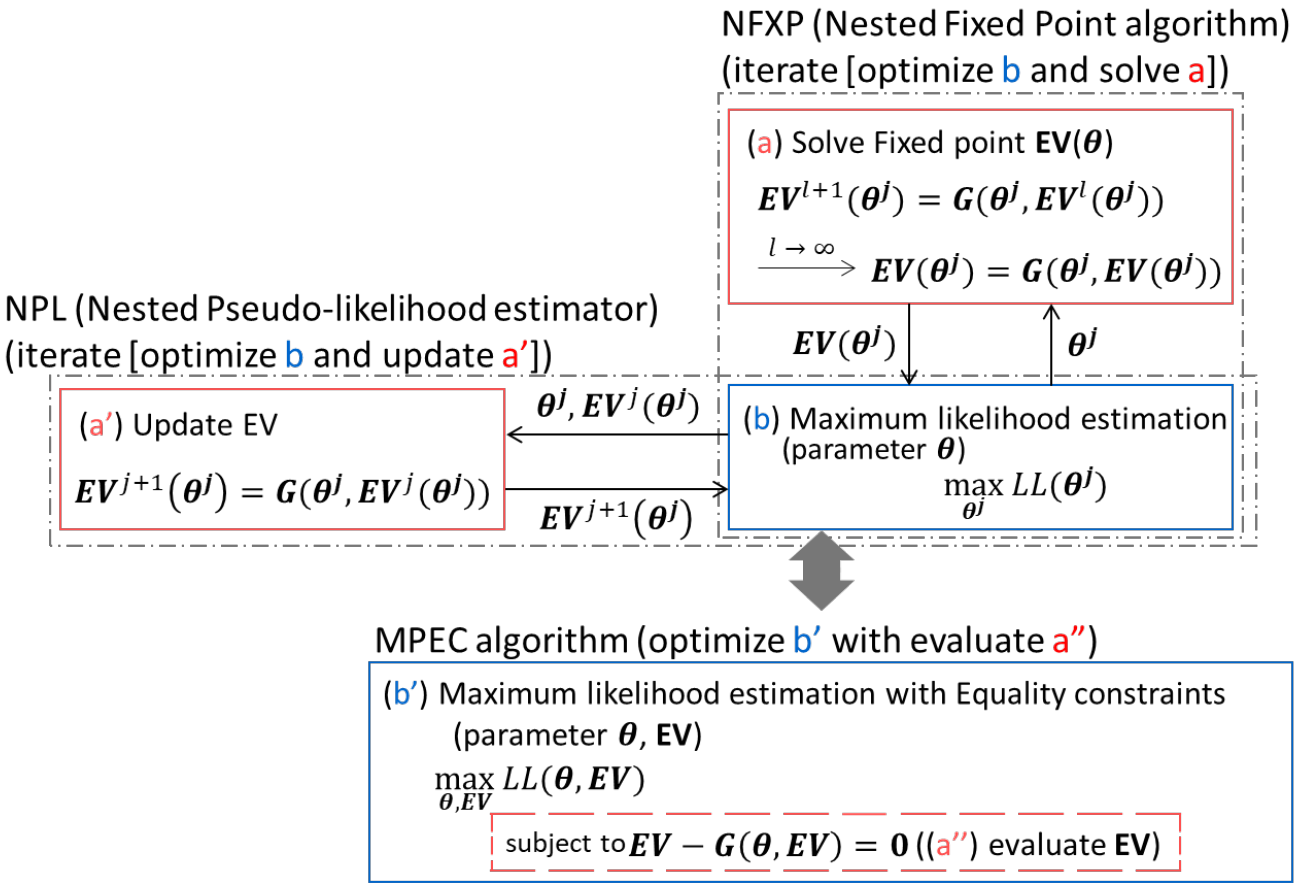
Figure 1: CPU times for MPEC-AMPL and NFXP-NK as a function of sample size





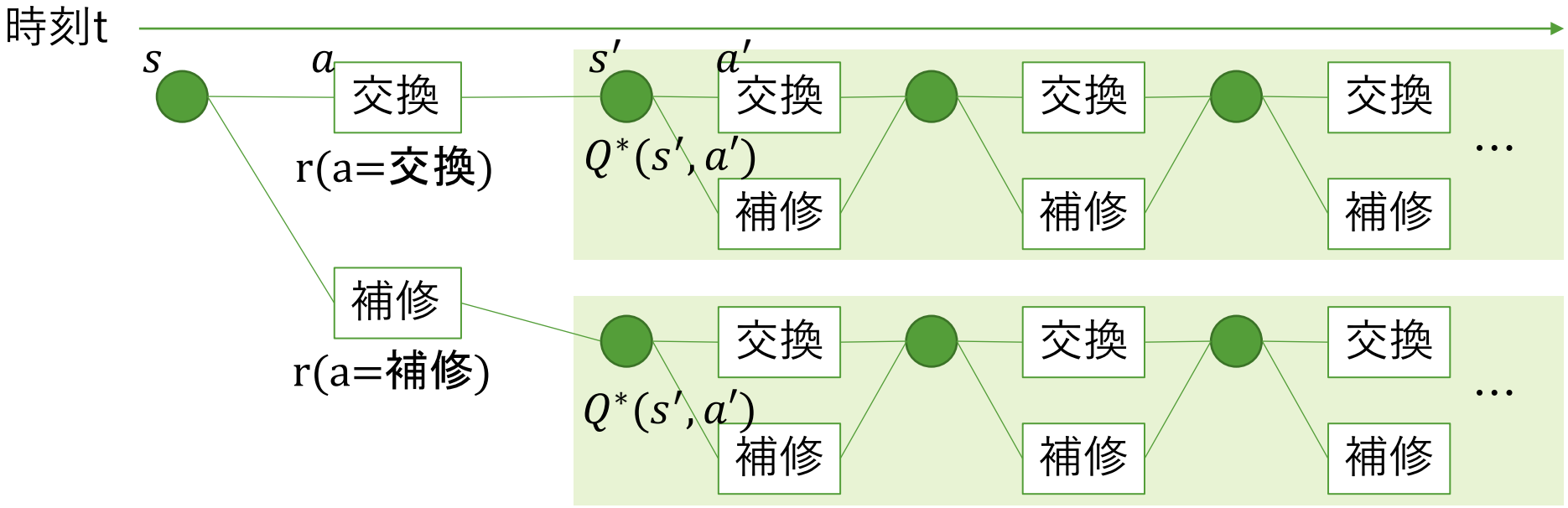
推定アルゴリズムの関係性

Urata & Hato, under review, 2023



強化学習／逆強化学習

最適行動価値関数



方策π：状態s→行動a

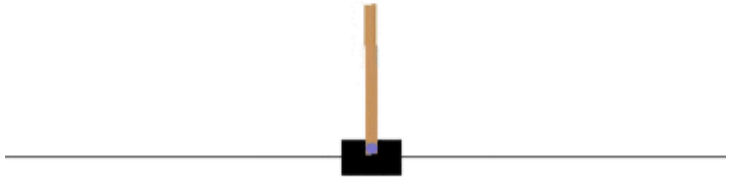
$$Q^*(s,a) = r + \beta \, max_{a'} \, Q^*(s',a')$$ ベルマン方程式

$$L(\theta) = E[Q(s,a) - (r + \beta \, max_{a'} \, Q(s',a'))]$$

β: 割引率

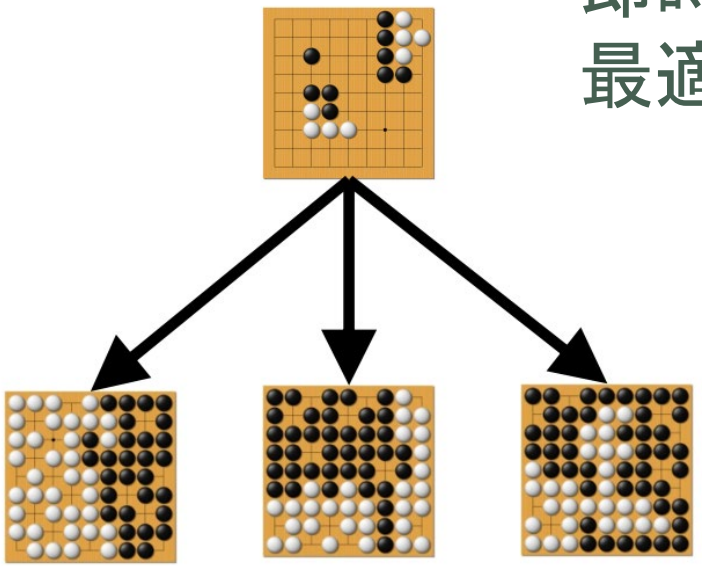
即時報酬rと強化学習

CartPole問題



https://blog.brainpad.co.jp/entry/2017/02/24/121500

囲碁



即時報酬rは固定／決まらない
最適行動価値関数Q*を決める

美添(2019)

推定アルゴリズム：Q値の直接学習と関数近似

Q-Learning:

α: 学習率

行動価値関数$Q(s,a)$のベクトルを更新していく

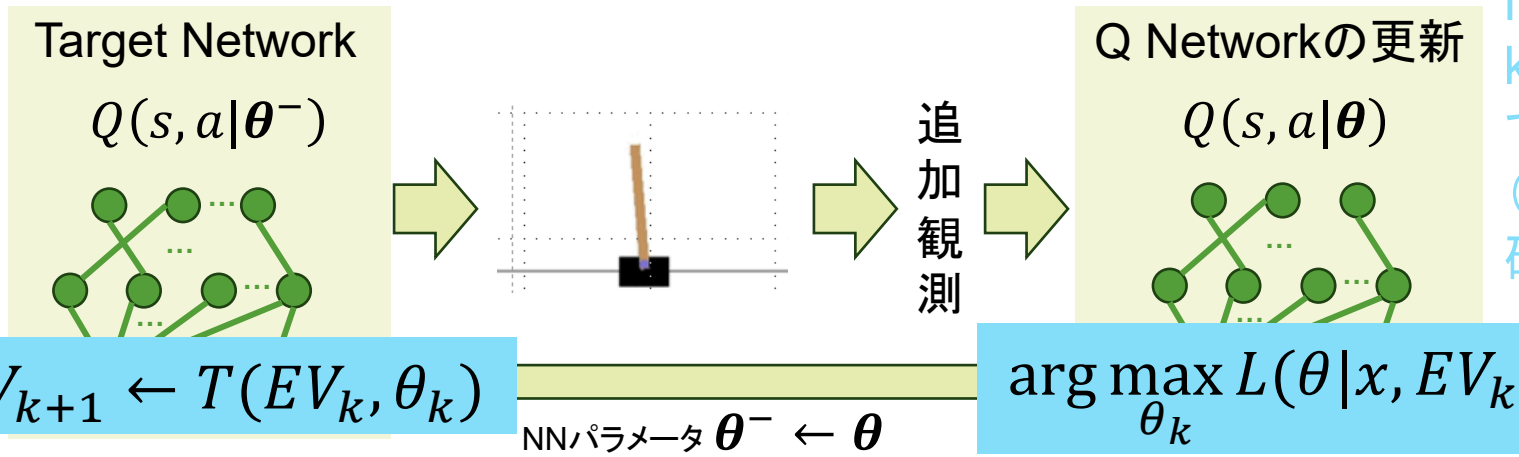
Qの探索≒MPEC型アルゴリズム（ただし制約条件はない）

更新過程

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \beta\, max_{a'}\, Q(s',a') - Q(s,a)]$$

状態・行動(s,a)の組数が多い・隣接状態間のQの差を小さいはず

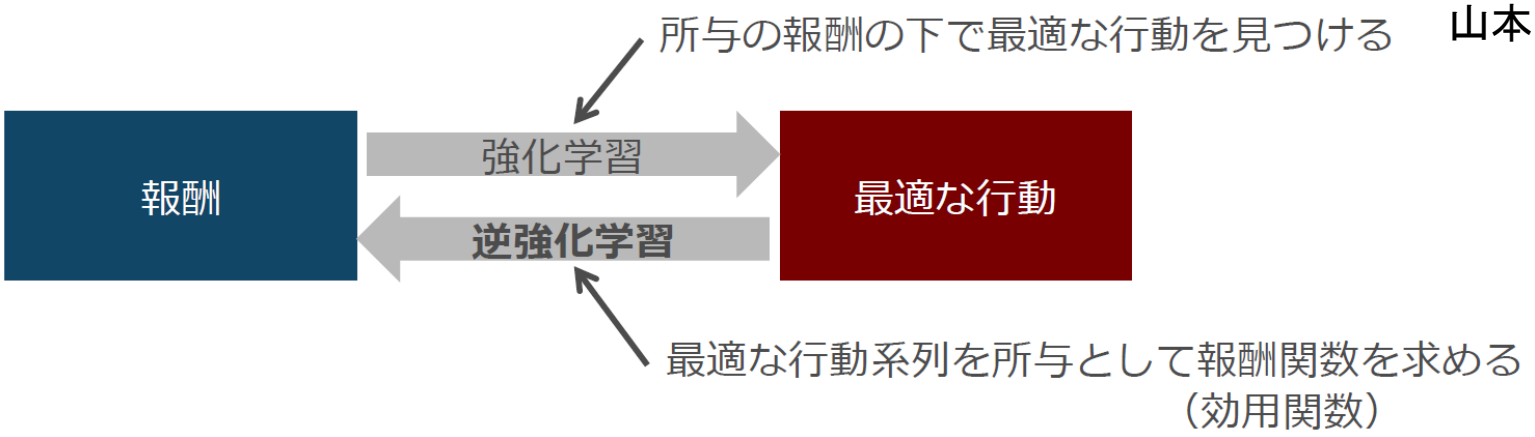
Q値を関数近似する（Sutton 1988）→ Deep Q-Network



NPL型でstep kのパラメータで将来効用EV（θ_k）を求め、確率計算

逆強化学習

山本(2019)



観測を最適な行動と仮定し、再現するための報酬rを求める
（推定した報酬関数により、エキスパートの行動再現を目指す）

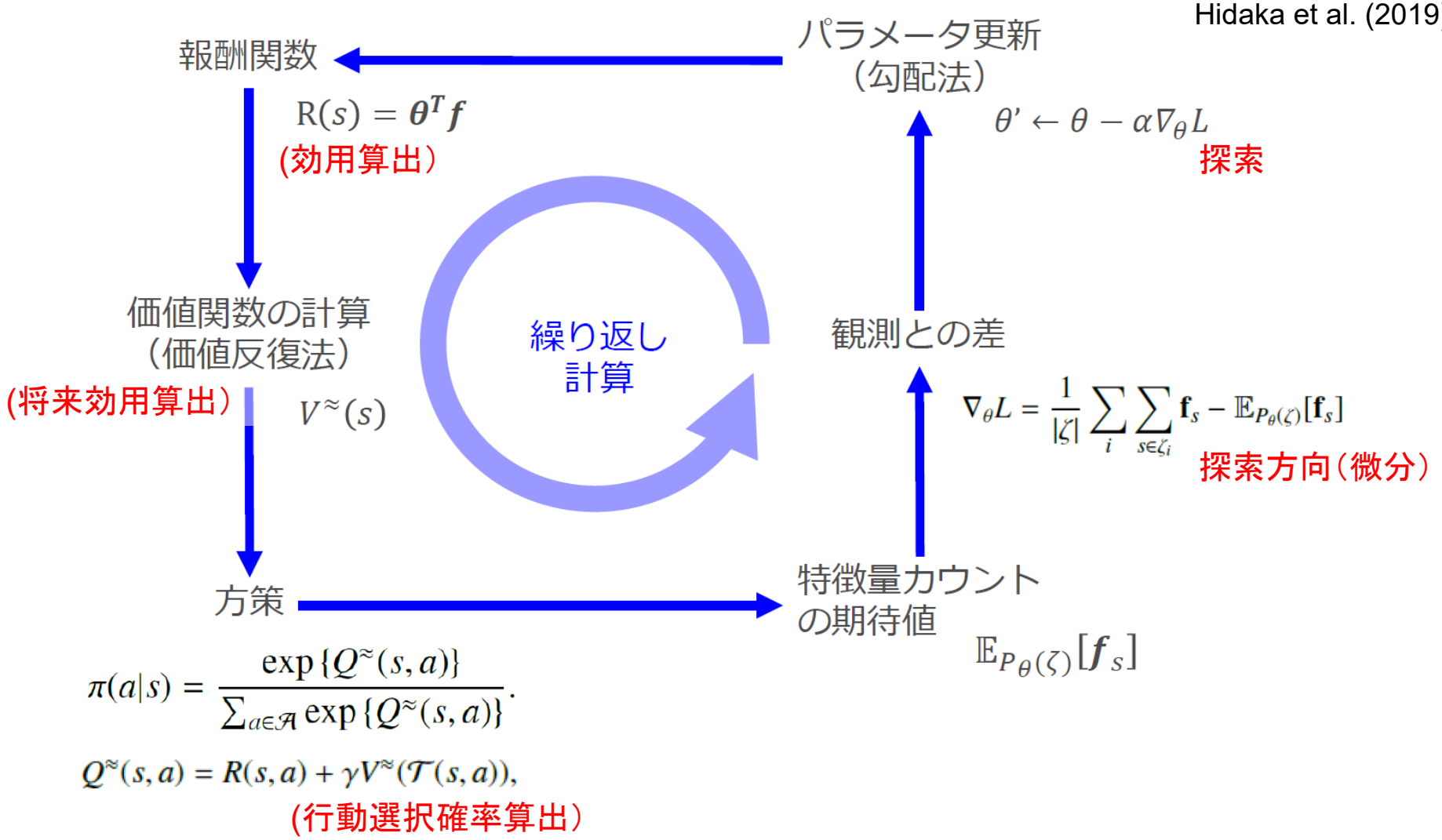
最大エントロピー逆強化学習：意思決定の不確実性を考慮

$$\max_R \sum_{s\in S} P^*(s) \sum_{a\in A} \pi^*(a|s)(R(s,a) - \log \sum_{a'\in A} \exp(R(s,a')))$$

最大エントロピー逆強化学習の推定

山本(2019)

Hidaka et al. (2019)



最大エントロピー逆強化学習の推定

中妻(2019)

価値関数の計算：価値反復法
　数値反復計算によりBellman最適方程式を求める方法
　強化学習分野では一般的な方法の一つ

Step1
全てのsにおいて価値関数$V^{\approx}(s)$の値を初期化
これを$V_0^{\approx}(s)$と表す.

Step2
全てのsに対して以下を実行する.

$$Q_k^{\approx}(s,a) = \sum_{s\prime} P(s|s',a)\{R(s',a) + \gamma V_{k-1}^{\approx}(s')\}$$

$$V_k^{\approx}(s) = \log \sum_a \exp\{Q_k^{\approx}(s,a)\}$$ ←―― 通常のMDPならMax

Step3
$|V_k^{\approx}(s) - V_{k-1}^{\approx}(s)|$の値が十分小さくなるまでStep2を繰り返す

NFXPのInnerアルゴリズム（不動点求解）？

$P(s|s',a)$：状態s'で行動aを取ったときに
状態sに遷移する確率

Take-home message

- ML/DL分野の開発のマンパワー＋汎化への姿勢は凄まじく、パラメータ推定手法・アイディアは刺激になる
- とはいっても、根本的にはやることは同じ（探索＋評価がある、探索方向が大事）であり、基本を理解しましょう
- 実装・導入のハードルは下がっていて、機会は広がっていて、行き来することで双方の開発に繋がる

ご清聴ありがとうございました

email: urata.junji.gf@u.tsukuba.ac.jp

LORE FC lab for resilient & future city