研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

【タイトル仮】
時系列データの汎化的予測モデル開発のための評価関数の推定

武藤 克弥 (Katsuya Mutoh)
u255018@st.pu-toyama.ac.jp

富山県立大学 大学院 電子・情報工学専攻 情報基盤工学部門

August 4, 2023

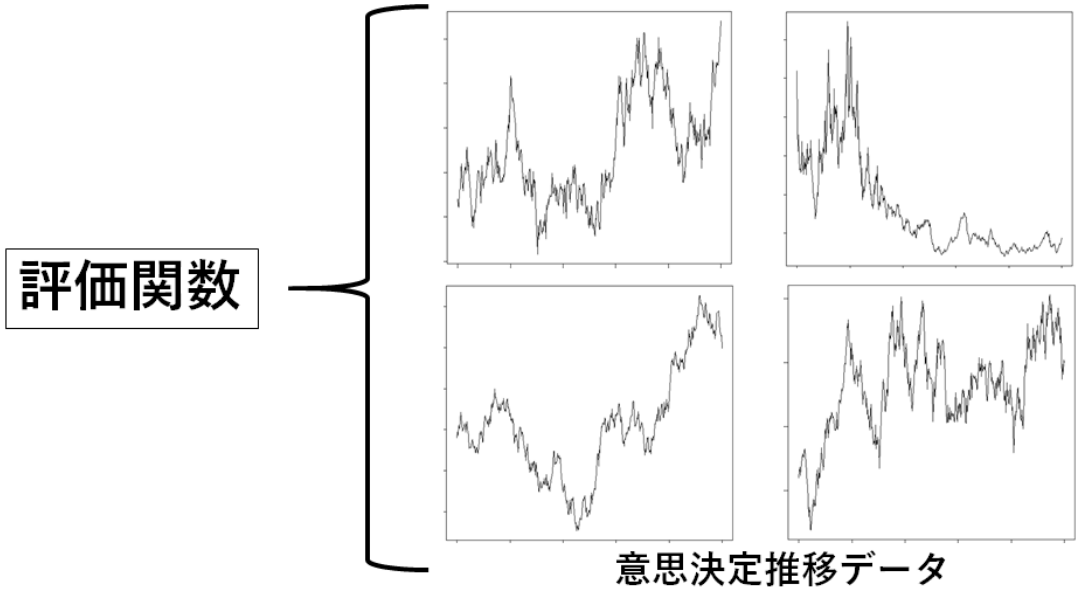
研究テーマについて

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

意思決定行動に関わる評価関数の推定

研究の背景 (仮説)

様々な意思決定活動はある評価関数で説明できるのではないか？
＝ 意思決定における行動は**ある評価関数をもとに決定される**



研究の目的

意思決定によって形成される時系列データ (金融・不動産・建設事業など) に対し，その結果の原因となった評価関数を推定するモデルを開発する．

研究テーマについて

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

評価関数推定のメリット

- 従来予測におけるデメリットの脱却
  - 従来の時系列予測にある「データの過学習」「パラメータ推定」などを行う手間がない
- 多分野への汎用性
  - 従来は少し分野が変わるだけで応用しにくいものもある
  - 高速道路の利用需要が上昇 ＝ 評価関数 V を最適化
    →新商品の売上予測

研究の新規性

意思決定モデルの拡張
→モデル (データ) に潜む評価関数を特定

研究テーマについて

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

研究の流れ

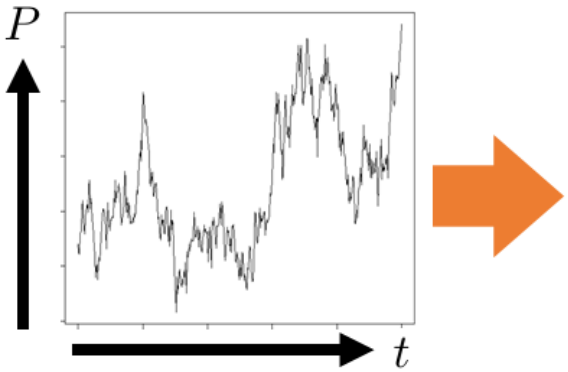
1. 意思決定モデルの構築　←現在取り組み中
2. 評価関数の推定手法の導入

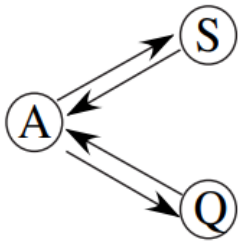
意思決定の枠組み

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

意思決定モデル [1]

- 時系列データに応じて，利益 (価値) の高くなる戦略を選択する





【仮定】
いずれかの戦略(アクティビティ)を選択

例) 建物の運用(3種のアクティビティ)
A: 運用継続
S: 運用休止
Q: 更地化(解体)

時刻 t でどの戦略を取っているのが最適なのか？
→最適値関数で評価

P: 時刻 t の状態 (価格や需要量)

確率微分方程式 (幾何ブラウン運動)

$$dP(t) = \alpha(t,P)dt + \sigma(t,P)dZ(t)$$

- $t \in [0,T]$: 意思決定期間
- $\alpha(t,P) = \alpha P$
- $\sigma(t,P) = \sigma P$
- α: ドリフト係数 (期待収益率)
- σ: 拡散係数 (ボラティリティ)
- $Z(t)$: ノイズ (1 次元 Wiener 過程)

[1] 長江, 赤松., 2004.

アクティビティ価値の導出

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

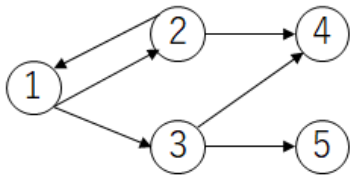
最適値関数: $V(t, P, n) = V_n(t, P)$

- 各時間 $t \in [0, T]$ で全アクティビティの価値を計算
  →時間 t でどの戦略を選べば最適か分かる

$V(t, P, n)$: 時刻 t におけるアクティビティ n の将来価値

→3要素で構成 (1) 利潤: $\pi_n(t, P)$, (2) 満期利潤, (3) 推移コスト: $C_{n,m(t)}$

- $n \in \{1, 2, 3, 4, 5\}$: アクティビティ
- r: 割引率



アクティビティ価値の導出

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

アクティビティ価値の導出 [2]

- 同時に全アクティビティ価値は求められない
  →グラフを分解して 1 つずつ求める

全アクティビティ導出手順 $V(t,P,n) = V_n(t,P)$

(1)終端アクティビティの価値を求める

これ以上遷移しないアクティビティ=④, ⑤

線形偏微分方程式

$$\mathcal{L}_{n'} V_{n'}(t,P) + \pi_{n'}(t,P) = 0 \quad \forall t \in [0,T)$$

終端条件: $V_{n'}(T, P(T)) = F_{n'}(P(T))$

偏微分作用素: $\mathcal{L}_n = \frac{\partial}{\partial t} + \frac{\partial}{\partial P}\alpha_n(t,P) + \frac{1}{2}\frac{\partial^2}{\partial P^2}\{\alpha_n(t,P)\}^2 - r$

(2)推移先が "終端アクティビティのみ" のアクティビティの価値を求める

→④, ⑤をもとに③の価値を求められる

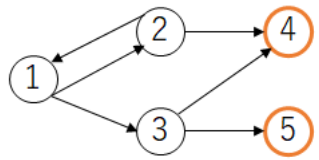
Fukushima型 merit関数による解の更新[2]

(3)サイクル構造内のアクティビティの価値を求める

→③, ④ (①⇔②の推移先)をもとに①, ②の価値を計算

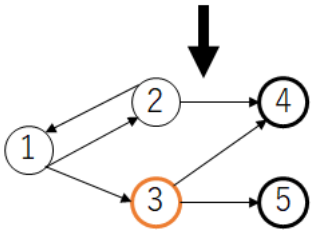
解の更新(2)と同様

(1)



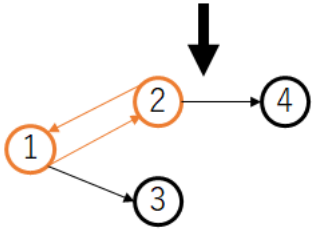
$V_4(t,P), V_5(t,P)$ for $0 \sim T$

(2)



$V_3(t,P) \leftarrow V_4(t,P), V_5(t,P)$ for $0 \sim T$

(3)



$V_1(t,P)$
$V_2(t,P)$ $\leftarrow V_3(t,P), V_4(t,P)$ for $0 \sim T$

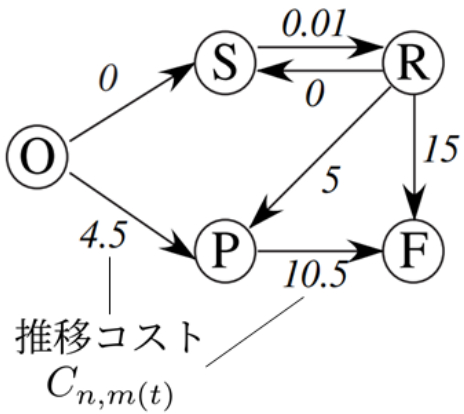
[2] Fukushima., 1992.

現在の取り組み (概要)

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

意思決定モデルの再現 [3]

- 有料道路の建設・運用事業における意思決定モデル



アクティビティ: $n \in \{O, S, R, P, F\}$

$V_n(t, P)$

(1) 利潤: $\pi_n(t, P)$, (2) ~~満期利潤,~~ (3) 推移コスト: $C_{n,m(t)}$

O: 事前評価 $\pi_{\mathrm{O}}(t, P) = -M_{\mathrm{O}}$

S: 計画凍結 $\pi_{\mathrm{S}}(t, P) = 0$

R: 再評価 $\pi_{\mathrm{R}}(t, P) = -M_{\mathrm{R}}$

P: 1区間供用 $\pi_{\mathrm{P}}(t, P) = X_{\mathrm{P}}P - E_{\mathrm{P}}$

F: 2区間供用 $\pi_{\mathrm{F}}(t, P) = X_{\mathrm{F}}P - E_{\mathrm{F}}$

$T = 20,\ r = 5\%,\ \alpha = 1\%,\ \sigma = 40\%$

$M_{\mathrm{O}} = 0.02,\ M_{\mathrm{R}} = 0.01$

$X_{\mathrm{P}} = 0.5,\ E_{\mathrm{P}} = 0.6,\ X_{\mathrm{F}} = 1,\ E_{\mathrm{F}} = 1$

[3] 長江, 赤松., 2004.

現在の取り組み[4]

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

意思決定モデルの再現

- アルゴリズムをプログラム化し，同じ結果が得られるようにする

(0) 時系列(シミュレーションデータ)の準備

幾何ブラウン運動(確率微分方程式)
→株価変動モデルに使用

$$dP(t) = \alpha P(t)dt + \sigma P(t)dZ(t)$$

$t \in [0, 20],\ r = 5\%,\ \alpha = 1\%,\ \sigma = 40\%$

解析解

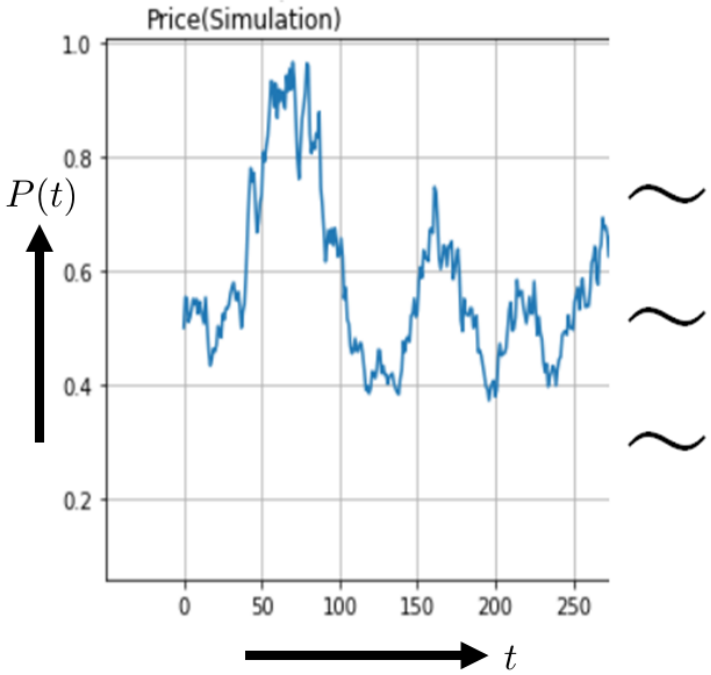
$$P(t) = P(0)\exp\left\{\left(\alpha - \frac{1}{2}\sigma^2\right)t + \sigma Z(t)\right\}$$

$Z(t)$: ノイズ (1 次元 Wiener 過程)

$Z \sim N(0, 1)$: 標準正規分布からの抽出

$\Delta T = 20/1000$: 時間幅

$P(0) = 0.5$: 初期需要



生成結果(URLのコード)

[4] Python でやってみた (Engineering)：モンテカルロ法, https://note.com/kiyo_ai_note/n/nfbe4c3f97e19

現在の取り組み

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

- 偏微分方程式を格子空間に差分 (離散) 化する

(1) 終端"F"の価値の導出

線形偏微分方程式

$$\mathcal{L}_n V_n(t,P) + \pi_n(t,P) = 0$$

終端条件: $V_{n'}(T,P(T)) = F_{n'}(P(T))$

偏微分作用素: $\mathcal{L}_n = \dfrac{\partial}{\partial t} + \dfrac{\partial}{\partial P}\alpha_n(t,P) + \dfrac{1}{2}\dfrac{\partial^2}{\partial P^2}\{\sigma_n(t,P)\}^2 - r$

偏微分方程式の差分化

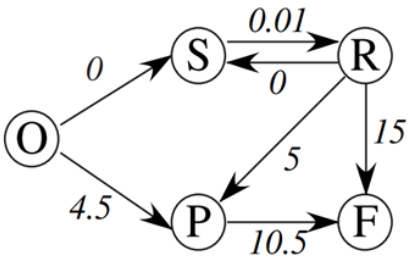
$$\mathcal{L}_n V_n(t^i,P^j) \approx \boldsymbol{L}_n^i \boldsymbol{V}_n^i + \boldsymbol{M}_n^i \boldsymbol{V}_n^{i+1}$$

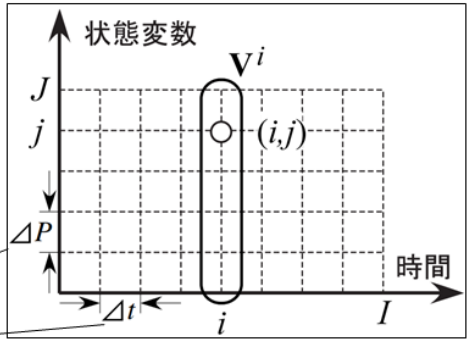
$$(t,P) \simeq (t^i,P^j) = (i\Delta T, P_0 + j\Delta P) \quad \begin{matrix}(i=0,1,\cdots,I)\\(j=0,1,\cdots,J)\end{matrix}$$

$I = J = 1000$

$\Delta T = \frac{20}{1000}$, $\Delta P = \frac{5}{1000}$:格子間隔

アクティビティ: $n \in \{O,S,R,P,F\}$





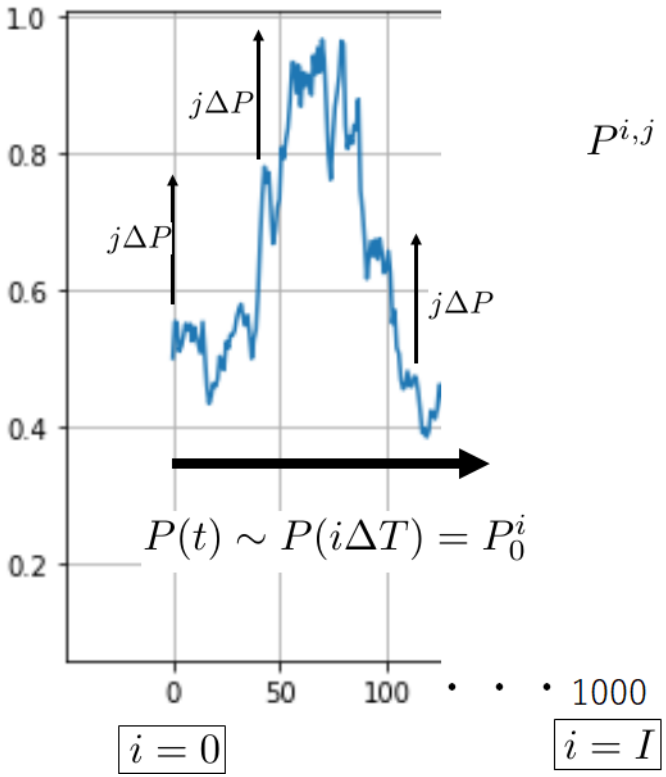
格子空間

現在の取り組み

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

(1) 終端アクティビティ"F"の導出

- (0) で求めた $P(t)$ を j(空間) 方向へ離散化



$$P^{i,j} = P_0^i + j\Delta P$$

$$= \begin{array}{|c||c|c|c|c|c|} \hline & j=0 & j=1 & j=2 & \cdots & j=J \\ \hline\hline i=0 & P_0^0 & P_0^0+\Delta P & P_0^0+2\Delta P & \cdots & P_0^0+J\Delta P \\ \hline i=1 & P_0^1 & P_0^1+\Delta P & P_0^1+2\Delta P & \cdots & P_0^1+J\Delta P \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline i=I & P_0^I & P_0^I+\Delta P & P_0^I+2\Delta P & \cdots & P_0^I+J\Delta P \\ \hline \end{array}$$

P_0^i

現在の取り組み

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

(1) 終端アクティビティ"F"の導出

- 期待収益率 $\alpha(t,P)$, ボラティリティ$\sigma(t,P)$、利潤 $\pi_n(t,P)$ も離散化

$$\boldsymbol{\alpha}(t,P) = \alpha P \simeq \boldsymbol{\alpha^{i,j}} (= \boldsymbol{\alpha}(t^i, P^j)) = \alpha(i\Delta t, P_0 + j\Delta P)$$

・$\boldsymbol{\alpha^{i,j}} =$

	$j=0$	$j=1$	$j=2$	$\cdots$	$j=J$
$i=0$	αP_0^0	$\alpha P_0^0 + \alpha\Delta P$	$\alpha P_0^0 + 2\alpha\Delta P$	$\cdots$	$\alpha P_0^0 + J\alpha\Delta P$
$i=1$	αP_0^1	$\alpha P_0^1 + \alpha\Delta P$	$\alpha P_0^1 + 2\alpha\Delta P$	$\cdots$	$\alpha P_0^1 + J\alpha\Delta P$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$i=I$	αP_0^I	$\alpha P_0^I + \alpha\Delta P$	$\alpha P_0^I + 2\alpha\Delta P$	$\cdots$	$\alpha P_0^I + J\alpha\Delta P$

・$\boldsymbol{\sigma}(t,P) = \sigma P \simeq \boldsymbol{\sigma^{i,j}}$ (同様)

・$\pi_n(t,P) \simeq \boldsymbol{\pi_n^i}(t^i, P^j) = P^{i,j} - \mathbf{1}$

$$= \begin{bmatrix} \alpha P_0^i - 1 \\ \alpha P_0^i + \alpha\Delta P - 1 \\ \alpha P_0^i + 2\alpha\Delta P - 1 \\ \vdots \\ \alpha P_0^i + J\alpha\Delta P - 1 \end{bmatrix} \qquad (n = F)$$

(1) 終端アクティビティ"F"の価値導出 (本手順)

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

$$\frac{\partial V_n}{\partial t} + \alpha(t,P)\frac{\partial V_n}{\partial P} + \frac{1}{2}\{\sigma(t,P)\}^2\frac{\partial^2 V_n}{\partial P^2} - rV_n + \pi_n(t,P) = 0$$

差分化 (Crank-Nicolson法)

$$\boldsymbol{V}_n^i = -(\boldsymbol{L}_n^i)^{-1}(\boldsymbol{M}_n^i \boldsymbol{V}_n^{i+1} + \boldsymbol{\pi}_n^i)$$

$$a^{i,j} = -\frac{\alpha^{i,j}}{4\Delta P} + \frac{(\sigma^{i,j})^2}{4(\Delta P)^2}, \qquad b^{i,j} = -\frac{1}{\Delta T} + \frac{(\sigma^{i,j})^2}{2(\Delta P)^2} - r$$

$$c^{i,j} = \frac{\alpha^{i,j}}{4\Delta P} + \frac{(\sigma^{i,j})^2}{4(\Delta P)^2}, \qquad d^{i,j} = \frac{1}{\Delta T} + \frac{(\sigma^{i,j})^2}{2(\Delta P)^2}$$

$$\begin{bmatrix} V_n^{i,0} \\ V_n^{i,1} \\ \vdots \\ V_n^{i,J-1} \\ V_n^{i,J} \end{bmatrix} = - \begin{bmatrix} b^{i,1} & c^{i,1} & 0 & 0 & 0 \\ a^{i,2} & b^{i,2} & c^{i,3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a^{i,J-1} & a^{i,J-1} & b^{i,J-1} \\ 0 & 0 & 0 & b^{i,J} & c^{i,J} \end{bmatrix}^{-1} \left(\begin{bmatrix} d^{i,1} & c^{i,1} & 0 & 0 & 0 \\ a^{i,2} & d^{i,2} & c^{i,3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a^{i,J-1} & d^{i,J-1} & c^{i,J-1} \\ 0 & 0 & 0 & a^{i,J} & d^{i,J} \end{bmatrix} \begin{bmatrix} V_n^{i+1,0} \\ V_n^{i+1,1} \\ \vdots \\ V_n^{i+1,J-1} \\ V_n^{i+1,J} \end{bmatrix} + \begin{bmatrix} \pi_n^{i,0} \\ \pi_n^{i,1} \\ \vdots \\ \pi_n^{i,J-1} \\ \pi_n^{i,J} \end{bmatrix} \right)$$

$(J$ 次元$)$ $\boldsymbol{L}_n^i (J \times J$ 次元$)$ $\boldsymbol{M}_n^i (J \times J$ 次元$)$ $(J$ 次元$)$ $(J$ 次元$)$

終端条件

$$\boldsymbol{V}_n^I = \boldsymbol{\pi}_n^i(t^I, P^j) = \begin{bmatrix} \alpha P_0^I - 1 \\ \alpha P_0^I + \alpha\Delta P - 1 \\ \alpha P_0^I + 2\alpha\Delta P - 1 \\ \vdots \\ \alpha P_0^I + J\alpha\Delta P - 1 \end{bmatrix}$$

$(n = F)$

$\leftarrow i = I$ から $i = 0$: 後ろ向きに V_n^i を求めていく

(2)(3) 残りのアクティビティ価値の導出 (本手順)

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

Merit 関数 [5]

- Merit 関数を実装

残りアクティビティ価値を求める問題

[NCP^i-c] Find $\boldsymbol{X}_c^i$ such that

$$\boldsymbol{X}_c^i \cdot \boldsymbol{G}_c^i(\boldsymbol{X}_c^i) = 0, \quad \text{and} \quad \boldsymbol{X}_c^i \geq \boldsymbol{0},\ \boldsymbol{G}_c^i(\boldsymbol{X}_c^i) \geq \boldsymbol{0}.$$

Fukushima型 merit関数

[Alg-Merit]

Step 0 初期可能解 $\boldsymbol{X}_c^{i(1)} \in \mathcal{R}_+$, $k := 1$.

Step 1 降下方向ベクトルの決定.

$$\boldsymbol{d}^{(k)} := \boldsymbol{H}(\boldsymbol{X}_c^{i(k)}).$$

Step 2 ステップ・サイズ α を，以下の一次元探索問題の解として求める.

$$\alpha^* = \arg.\min_{\alpha \in [0,1]} \Phi\left(\boldsymbol{X}_c^{i(k)} + \alpha \boldsymbol{d}^{(k)}\right).$$

Step 3 解の改訂. $\boldsymbol{X}_c^{i(k+1)} := \boldsymbol{X}_c^{i(k)} + \alpha^* \boldsymbol{d}^{(k)}$

Step 4 収束判定：収束していれば停止，そうでなければ $k := k + 1$ として Step 1 へ.

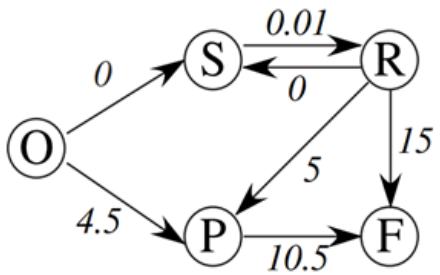
```python
# (2)を求めるMerit関数
def getGxFromViForNc(X, n):
    act_now[n][i_rev-1,:] = -(np.linalg.inv(L[i]) @ (X.T + M[i] @ act_now[n][i_rev,:].T
                                                     + act_pi[n][i_rev-1,:].T)).T
    On = list(act_trans[n])
    Gn_ks = []
    for ix in On:
        Gn_ks.append(act_now[n][i_rev-1,:] - act_now[ix][i_rev-1,:]
                     - np.matrix(np.full((1,J), act_cost[n][ix]), dtype='float'))
    return np.minimum.reduce(Gn_ks)

# (3)を求めるMerit関数
def getGxFromViForC(X, nc):
    act_now[nc[0]][i_rev-1,:] = -(np.linalg.inv(L[i]) @ (X.T + M[i] @ act_now[nc[0]][i_rev,:].T
                                                         + act_pi[nc[0]][i_rev-1,:].T)).T
    act_now[nc[1]][i_rev-1,:] = -(np.linalg.inv(L[i]) @ (X.T + M[i] @ act_now[nc[1]][i_rev,:].T
                                                         + act_pi[nc[1]][i_rev-1,:].T)).T
    On = list(act_trans[nc[0]])
    On2 = list(act_trans[nc[1]])

    Gn_ks = []
    for ix in On:
        Gn_ks.append(act_now[nc[0]][i_rev-1,:] - act_now[ix][i_rev-1,:]
                     - np.matrix(np.full((1,J), act_cost[nc[0]][ix]), dtype='float'))

    Gn_ks2 = []
    for iy in On2:
        Gn_ks2.append(act_now[nc[1]][i_rev-1,:] - act_now[iy][i_rev-1,:]
                      - np.matrix(np.full((1,J), act_cost[nc[1]][iy]), dtype='float'))
    Gn_ks2.append(np.minimum.reduce(Gn_ks))
    return np.minimum.reduce(Gn_ks2)
```

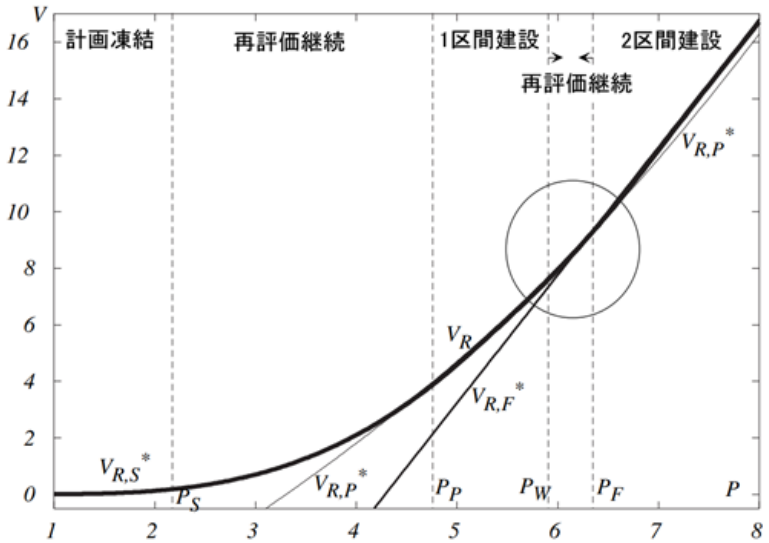
[5] Fukushima., 1992.

暫定結果

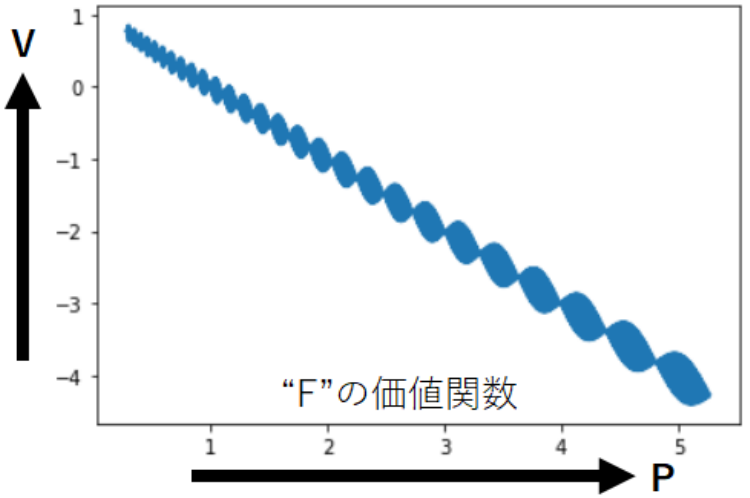
研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

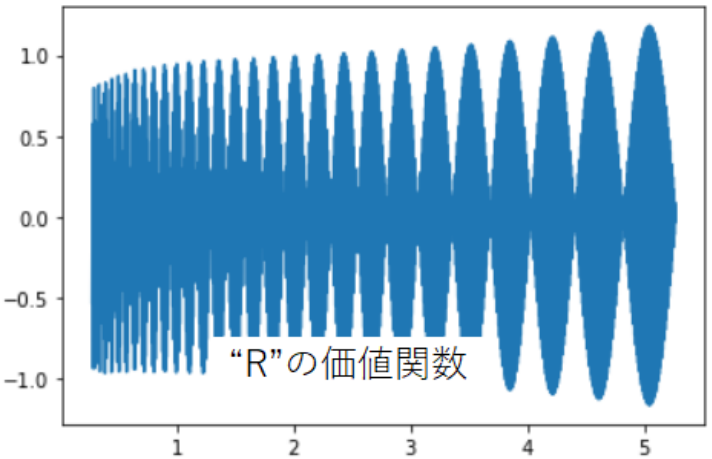
- $i = 1000 \sim i = 900$ までは振動、900 以前では発散 (nan 値) した





理想の出力イメージ t=15 (i= 750)





実際の出力(i= 999)

結果の分析

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

振動・発散の原因

- 状態空間 P の離散化が正しくない可能性 (P11)
- 偏微分方程式の差分化→行列化の過程を自力で行い分析 (P13)

差分化の例

$$\frac{\partial U}{\partial \tau} = \frac{\partial^2 U}{\partial x^2} \xrightarrow{\text{差分化}} \frac{U_{i+1,j} - U_{i,j}}{\Delta \tau} = \frac{U_{i+1,j+1} - 2U_{i+1,j} + U_{i+1,j-1}}{(\Delta x)^2}$$

行列化

$$\begin{pmatrix} 1+2r & -r & 0 & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+2r & -r \\ & & & -r & 1+2r \end{pmatrix} \begin{pmatrix} U_{i+1,j_{\min}+1} \\ U_{i+1,j_{\min}+2} \\ \vdots \\ U_{i+1,j_{\max}-2} \\ U_{i+1,j_{\max}-1} \end{pmatrix} = \begin{pmatrix} U_{i,j_{\min}+1} \\ U_{i,j_{\min}+2} \\ \vdots \\ U_{i,j_{\max}-2} \\ U_{i,j_{\max}-1} \end{pmatrix} + r \begin{pmatrix} U_{i+1,j_{\min}} \\ 0 \\ \vdots \\ 0 \\ U_{i+1,j_{\max}} \end{pmatrix}$$

今後の予定

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

今後の予定

- 振動・発散の原因を特定
- 評価関数推定の実装
- 研究の背景・目的を明確化

評価関数の推定

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

ロボットの起き上がり運動獲得のための
正規化ガウス関数ネットワーク (NGnet) を用いた Actor-critic 強化学習 [6]

研究概要

「目標出力 y(t) メートルまで頭が上がるように関節を動かす」ように学習させる

- ロボットは目標出力に達するように最適な「関節の動き」を探る
- 推定評価関数=「この動きをすれば**目標出力**に達する」と考えた自己評価の集約
  →この関数を最適化するような行動を行う

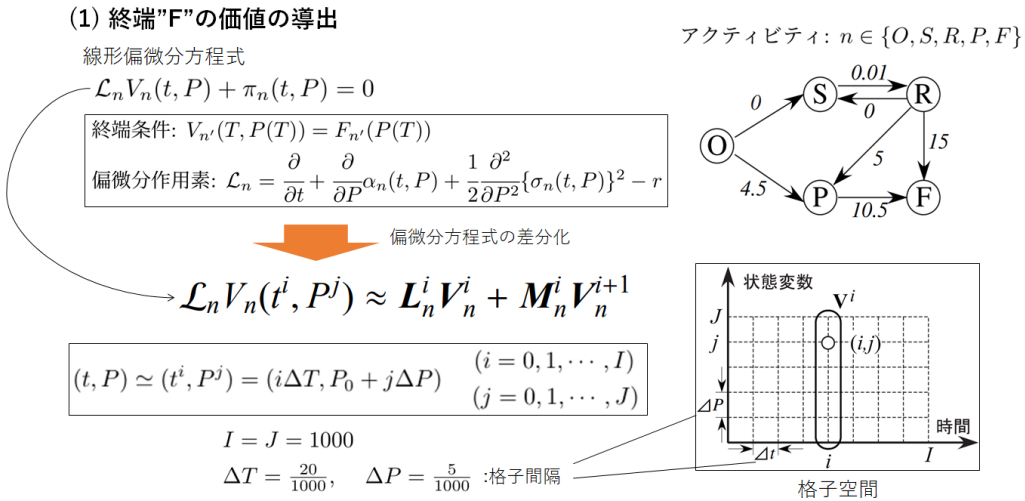


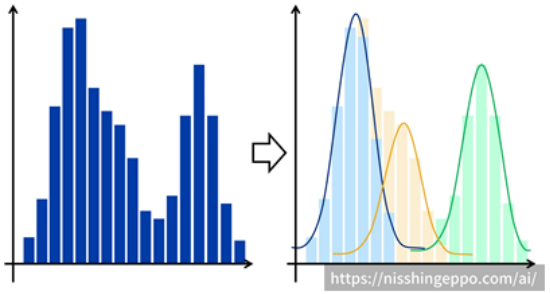
図 1: 推定評価関数と実際の行動

[6] 森本, 銅谷., 1999.

評価関数の推定

研究概要
意思決定モデル
進捗状況
結果
結果
今後の予定
評価関数の推定

評価関数推定の概要



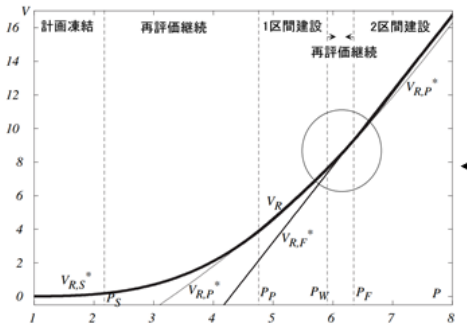
(1)関数の設定

利潤: $\pi_n(t,P)$, 推移コスト: $C_{n,m(t)}$

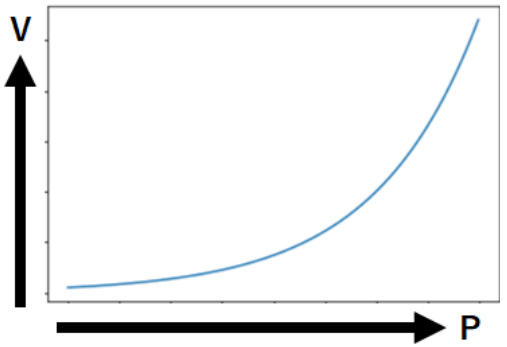
$\longrightarrow$ $b_{1,k}(\boldsymbol{x})$ $b_{2,k}(\boldsymbol{x})$

(2)アクティビティ価値導出アルゴリズム
(前々ページ)

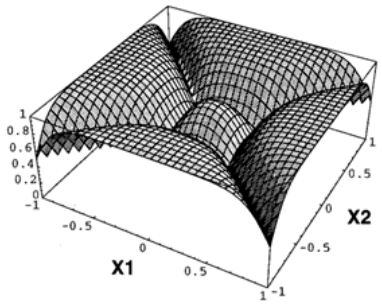
$$y(\mathbf{x}) = \sum_{k=1}^{K} w_k \underbrace{b_k(\mathbf{x})}_{\text{基底関数}}$$



2乗誤差e



V(価値)-P(状態)図の出力



(5)評価関数の出力

(3)eが最小になるように $\boldsymbol{w_k}$ を更新

(4) (1)～(3)を指定回数だけ繰り返す