

# 生命保険アクチュアリー業務における強化学習の応用

XXXXXX 生命

ExplPiP1EO

2019/11/25 : 日本アクチュアリー会 2019 年度年次大会

## ■概要

- ・近年、DQN・アルファ碁の成功を皮切りに、機械学習の一分野である**強化学習**の研究が活況を見せている。アクチュアリー界隈においてはデータサイエンスの一環として機械学習への注目が集まりつつあるが、教師あり学習・教師なし学習といった比較的標準的な分野に限られており、強化学習の扱いは極めて小さい。
- ・本論文では、**生命保険アクチュアリー**の基本業務のいくつかについて、極めて簡略化したモデルに強化学習を適用した場合に得られる結果及び問題点について考察する。アルゴリズムは深層強化学習において標準的なものに限定し、アクチュアリーの専門知識を使わずにどの程度の結果が自動的に得られるのか、というスタンスを取ることにした。
- ・本論文で扱った題材は非常に基礎的なものばかりであるが、それでも強化学習の標準的なアルゴリズムをそのまま適用するだけでは安定して効率的に解くことはできなかった。これはアクチュアリーが普段扱う問題は既に解析解が存在するものや既存のソルバーで高精度で解けるものが多く、強化学習が方法論的に不適であったり、現行の標準的なアルゴリズムでは解が不安定になるなどの理由による。より安定して解ける方法を模索するか、あるいは新たな領域を探すかは今後の課題であると思われる。

## ■1.導入

- ・強化学習は環境との相互作用という試行錯誤を通じて、優れた行動戦略を自律的に学習・生成する機械学習の一分野であり、ある意味最もAIらしいAIの1つである。近年、DQN(1)・アルファ碁(2)の成功を皮切りにその研究は非常な活況を見せている。次々と新しいアルゴリズムが提案される中で、限られてはいるものの現実世界での応用も散見されるようになってきた。
- ・強化学習の論文件数はarXiv<sup>1</sup>において2014年以前にはそれまでの累計で409件であったものが2015年には年間110、2016年には285、2017年には665、2018年には1376になり、2019年においては11月3日現在まで2332件になるなど爆発的に増加している（累計5177件）。
- ・アルゴリズムも価値ベースのものから方策ベース・進化的方法・モデルベース、解くべき問題も単純なMDPからマルチタスク・階層タスク・マルチエージェント・POMDP、学習の枠組み自体も行動学習から模倣学習・逆強化学習に亘るなど多彩な広がりを見せている。
- ・応用が期待される分野としてもデータセンターの電力制御・ニューラルネットの最適構造の生成・言語タスクにおける流用・ロボティクス・ゲームのテストプレイ・タクシーの配車・広告配信・化学物質の生成・制震構造の開発・量子コンピュータの回路最適化など多岐に亘っている。

## ■先行研究

- ・このように活況を見せている強化学習の研究界隈ではあるが、**保険分野**におけるものは限られている。直近のものとして、**住宅保険の更新価格**について強化学習を適用した研究(3)があるものの、それ以外となると近年のものはあまり見当たらない。また、この研究もkクラスタリングにより生成した特徴量に基づく比較的古典的なものであり、ディープラーニング等の深層強化学習の技術は用いられていない。

## ■趣旨

- ・本論文は、**生命保険アクチュアリー業務**のうちの基本的なものの幾つかに対して**深層強化学習**を適用した結果について報告する。既に述べたように保険分野における先行研究は非常に限られ

<sup>1</sup> <https://arxiv.org>

ており、本論文でも高度なことを行っているわけではないが、生命保険アクチュアリー業務における複数分野に亘る対象を扱ったものは本論文が恐らく初めてである。

## ■概要

・第1節では**保険料計算**を扱う。単純なYRT・更新を考慮したYRTのプライシングに強化学習を適用した場合について述べる。第2節では**資産運用**を扱う。単純な資産価格モデルに基づく静的な資産配分問題・デュレーションマッチングについて述べる。第3節では**責任準備金**を扱う。強化学習の標準的な方法では上手く行かないことを示し、これを解決するための幾つかの試みについて述べる。最後に、付録として**ディープラーニングフレームワークを用いた会社価値の詳細感応度の自動導出・破産事象における事後分布の導出**について紹介する。

・論文全体を通して、深層強化学習の領域において標準的とされるものをできるだけそのまま使用するように努めた。つまり、アクチュアリーの専門知識を用いずにどの程度のことが自動的にできるか、というスタンスを取っている。敢えてこのように制限することで、アクチュアリーの専門知識を活用すべき部分・今後伸ばしていくべき部分が何処にあるかが浮き彫りになるとを考えた。

## ■技術的注意事項

・ディープラーニングを多用する関係上、扱う**数値のオーダー**（保険料の大きさなど）は、ディープラーニングで学習しやすい範囲に限ることにした（±10程度）。このため、数値のオーダーとしては不自然なものが多い。実際のデータや問題に適用する場合には数値の標準化などの処理が別途必要である<sup>2</sup>。

・同様に、問題が解けるかどうかに集中したかったため、**数理モデルは非常に単純化**している。  
・本論文では強化学習を用いて種々の最適化を検討するが、**最適化の目的関数は利益あるいはその現在価値に限定**した。保険会社経営の立場からはリスク管理・営業員ターンオーバーの改善・解約率の改善・ブランド管理・商品ラインナップの充実など様々な目的が考えられるが、本論文では一切取り扱わない。

・全ての事象をシミュレーションで直接表現した。元データなどからの統計的推定あるいは機械学習・モデリングなどは一切扱わず、**アサンプションは既知**とした<sup>3</sup>。

---

<sup>2</sup> どのように標準化すべきか、というのは将来的なテーマになるかもしれない。例えば会社が成長段階なのか停滞段階なのかによっても変わってくると思われる。実装上は、標準化の必要性自体はニューラルネット側の事情であるためニューラルネットそのものに組み込むのが妥当と思われる。しかし、どう標準化すべきかは環境によって数値範囲が異なるので環境側に組み込むしかないという面もある。環境側に対ニューラルネット用のIOを別途組み込んだり、ラッパーオブジェクトを介する方法もあるが、API設計をうまくやらないと分析が面倒である。今回試した限りでは、ニューラルネットにおけるバッチ正規化・スケーリングレイヤーの適用はあまり良い結果とはならなかった。

<sup>3</sup> 「シミュレーションできる」という意味であり、「強化学習エージェントがこの情報を直接使用できる」という意味ではない（強化学習に馴染みがないと、この意味は少し分かりにくいかかもしれない）。勿論、この仮定は特に需要関数や解約率においては非現実的であり、この仮定の信頼度に応じて最適解が（例えばベイズ的に）どのように分布するか、あるいはロバスト性がどの程度あるか、などは別途検証すべきテーマだと思われる。

・本研究は全て **Python** で行った。また、フレームワークは一部を除き全て **Keras** を用いた<sup>4</sup>。アルゴリズムは **DQN**・**A2C**<sup>5</sup>・**DDPG**<sup>6</sup>を主に用い、一部に **NAF**<sup>7</sup>・**AWR**<sup>8</sup>・**ES**<sup>9</sup>・**GA**<sup>10</sup>を用いた。活性化関数は全て **LeakyReLU**・オプティマイザは全て **Adam** を用いた<sup>11</sup>。ハイパーパラメータのチューニングはグリッドサーチで多少行っている程度である。このため、アルゴリズム間の優劣についてはあまり追求していない。特に AWR に関しては実装が最後だった関係で詰めている。NN は DDPG を除き全て(128, 64, 32)の中間ニューロン数とした。DDPG も同程度の規模である。サンプル数は断りのない限り全て **64 万ステップ**相当である。

・本論文における Python コードと Jupyter Notebook は全て以下の執筆者 GitHub から **ダウンロード可能**とする予定である。実装の詳細はそちらを参照されたい。

<https://github.com/ExplPiP1E0/RLInBasicLIActuarialWork2019>

・尚、本論文の内容は全て執筆者の個人的見解に基づくものであり、所属する組織の見解を示すものではない。

---

<sup>4</sup> サンプリングを高速化するために環境は全て NumPy の一括計算ベースで実装した。このため、通常の強化学習フレームワークとは若干 API 等が異なっている。具体的にはエージェント数 64 の一括計算が基本となっている。これは単純に執筆者の実装能力では分散・並列処理が難しかったためである。尚、Keras 以外を用いたのは第 1 節における **ChainerRL** と付録における **PyTorch** のみである。また、自作コードの性能検証のために、CartPole で ChainerRL との比較を行い同程度であることを確認している。

<sup>5</sup> Advantage Actor Critic (6)

<sup>6</sup> Deep Deterministic Policy Gradient (7)

<sup>7</sup> Normalized Advantage Function (10)

<sup>8</sup> Advantage Weighted Regression (11)

<sup>9</sup> Evolution Strategy (8)

<sup>10</sup> Genetic Algorithm (9)

<sup>11</sup> 深層強化学習の研究で Adam が使用されることはあるが、本論文の範囲では Adam が最も性能が良かったため、Adam としている。

## ■2.技術的背景

- ・本節では本論文のメイントピックである深層強化学習について概説する。詳細については別途専門書をあたって頂きたい<sup>12</sup>。アクチュアリー的には「EVを、自分自身の行動まで含めたファクターの再帰式を解くことで最適化する」と見なしてしまってもひとまずは構わない。

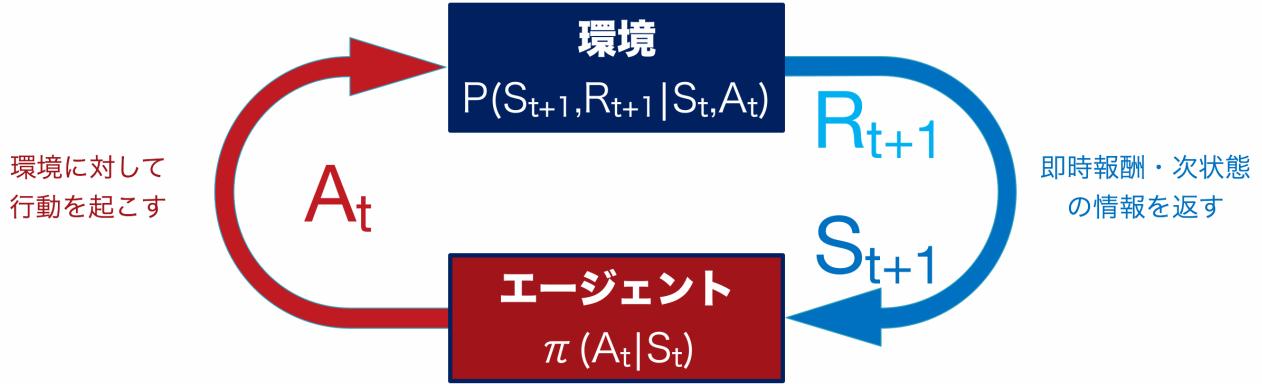
## ■強化学習

- ・強化学習は機械学習三大分野の1つであるが、他2つ（教師あり学習・教師無し学習）に比べるとややマイナーで、機械学習・ディープラーニングを解説した書籍などでも除外されていることが多いようである。これは数理的な構造が他2つと大きく異なり更には高難度であること、標準化が難しく単純なライブラリで対応できないことが多いこと、実用可能なケースが未だ模索段階で実需要が少ないとこと、必要な計算量が非常に多いこと、莫大なデータあるいは実用的なシミュレータが必要になること、安定性や再現性に問題を抱えており使いこなしが未だ難しいこと、などが理由であると思われる。本項では強化学習の問題設定や標準的な概念について簡単に説明する。強化学習のアルゴリズムには非常に多くの種類があるが、ここでは最も単純なQ学習についてのみ説明する。このため若干不正確な書き方になっているが了承して頂きたい。

### ◇MDP

- ・強化学習の問題設定はマルコフ決定過程（Markov Decision Process : MDP）と呼ばれる。これは、将来の行動を決定するには現在の状態だけが分かっていれば十分で、過去の状態や経緯について知っていても何の役にも立たない、という仮想的な問題設定である。MDPの基本要素は、「状態・行動・推移規則・即時報酬・方策」であり、このうち「状態・行動・推移規則・即時報酬」を環境（environment）と呼ぶ。
- ・状態（state）とは文字通り環境の様々な状態のことである。計算に使用可能なデータという側面を強調するために観測量と呼ぶことも多い。
- ・行動（action）とは、各々の状態において実行可能な選択肢あるいは実行された行動のことである。
- ・即時報酬（reward）とは、行動を起こした際に即時に獲得できる報酬のことである。
- ・推移規則とは、ある状態においてある行動を取ったら、どれだけ即時報酬を得てどの状態に移行するかという法則あるいはルールのことである（この名称は一般的ではない）。
- ・方策（policy）とは行動のルールあるいはパターンのことである。「この状態だったらこの行動をする」というルールを、ゴールを除く全ての状態に対して定めたものである。
- ・MDPでは「ある状態において、行動を起こすと、推移規則に従い即時報酬をもらいつつ次の状態へ移行する」というパターンを繰り返していくことになる。この「状態、行動、即時報酬、次の状態へ推移」のワンセットを1ステップと呼ぶ。最初の状態を0ステップ目・2番目の状態を1ステップ目…のように呼び、各々 $S_0, S_1, \dots, S_t, S_{t+1}$ のように書く。同様に、各々のステップの行動を $A_0, A_1, \dots, A_t$ 、即時報酬を $R_1, R_2, \dots, R_t$ のように書く（即時報酬のみ0ではなく1からスタートすることに注意する。これは、行動した結果として得られるというニュアンスがあるため、次状態への推移と同列で扱われているためである。但し、このあたりは書籍にも依る）。方策は $\pi$ で表す。
- ・MDPには終わりがあるタイプと終わりがないタイプがあり、前者をエピソード的（episodic）、後者を連続的（continuing）と呼ぶ。エピソード的な場合、開始から終端までの一区切りをエピソードと呼ぶ。更に、スタートに相当する状態を開始状態（start state）、ゴールに相当する状態のことを終端状態（terminal state）と呼ぶ。
- ・行動の主体としてエージェント（agent）というものを設定し、上述の一連の時間発展をエージェントと環境との相互作用として表現することが多い。

<sup>12</sup> 全般については(12)を、理論面詳細については(13)を、深層強化学習のアルゴリズム全般については(14)を、実装全般に関しては[15]と[16]を参照されたい。



### ◇価値関数

・強化学習における目的関数に相当するものが**(行動) 価値関数**  $Q_\pi$ である。これは現在の状態  $s$ において行動  $a$ を取り、かつその後に方策  $\pi$  に従った場合に得られる即時報酬の割引現在価値の期待値であり、概ね EV に近いものである。将来の即時報酬は方策  $\pi$  の影響を受けるので、価値関数はどの方策を用いているかを明示している。以降に方策  $\pi$  でなく最適行動を取り続けた場合に得られる即時報酬の割引現在価値の期待値を**最適行動価値関数**  $Q_*$ と呼ぶ。**Q 学習**はこの  $Q_*$  を求めることを通して最適方策を求める方法である（下記の式で  $G_t$  は収益と呼ぶが、会計収益と混同するので本論文では**利得**と呼ぶ）。

$$Q_\pi(s, a) := E_\pi[G_t | S_t = s, A_t = a] = E_\pi\left[\sum_{\tau=t} \gamma^{\tau-t} R_{\tau+1} | S_t = s, A_t = a\right]$$

$$Q_*(s, a) := \max_{\pi} Q_\pi(s, a)$$

### ◇ベルマン方程式

・多くの場合、強化学習アルゴリズムの基礎となる方程式が以下の**ベルマン方程式**である。これは、最大化演算を除けば生保数理における**ファクラーの再帰式**と同様のものである（あるいは、EV の再帰式と考えても良い）。

$$Q_*(s, a) = E[R_{t+1} + \gamma \max_{a_{t+1}} Q_*(S_{t+1}, a_{t+1}) | S_t = s, A_t = a]$$

### ◇関数近似

・状態行動空間が大きい場合には全ての  $s, a$  に対して  $Q_*$  を直接指定することはできないので、 $Q_*$  を  $s, a$  の特定のパラメータ付き関数（例えば  $w_1 \cdot s + w_2 \cdot a$  など）として表現することにして、パラメータを調整することで  $Q_*$  を近似的に表現する手法が取られる（**関数近似**）。このとき、以下のように、目的関数  $E(\mathbf{w})$  を定義して、これに対して勾配降下法を適用することで  $\mathbf{w}$  を上手く調整して  $Q_*$  をよく表現する～ベルマン方程式が概ね成り立つ～ $Q_*$  を得ることができる。

$$E(\mathbf{w}) := \left( R_{t+1} + \gamma \max_a Q_*(S_{t+1}, a) - Q_*(S_t, A_t) \right)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = -2 \left( R_{t+1} + \gamma \max_a Q_*(S_{t+1}, a) - Q_*(S_t, A_t) \right) \frac{\partial Q_*(S_t, A_t)}{\partial \mathbf{w}}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} + 2\alpha \left( R_{t+1} + \gamma \max_a Q_*(S_{t+1}, a) - Q_*(S_t, A_t) \right) \frac{\partial Q_*(S_t, A_t)}{\partial \mathbf{w}}$$

## ■ディープラーニング

・ディープラーニングは近年の計算機の高速化・データ量の増大・幾つかの技術的ブレイクスルーを背景に、主に画像・音声・文章の分野において華々しい成果を上げている技術である。

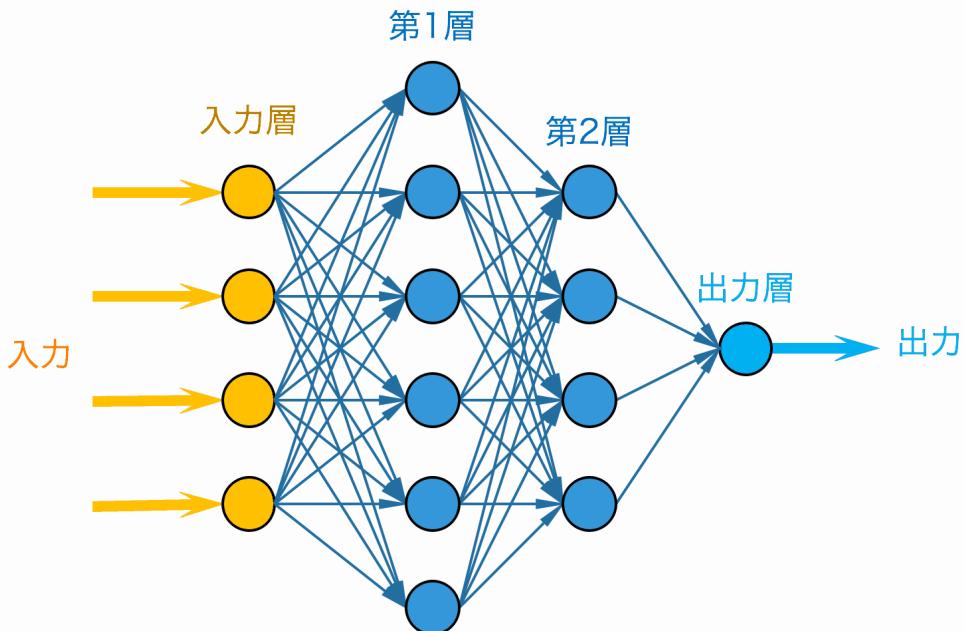
・基本的にはアフィン変換+非線形変換（活性化）をワンセットにした**ニューロン**・ニューロンを並列化した**レイヤー**を基本要素として、これらを組み合わせ、複層化することで合成関数的にあ

る特定の関数セットを作成する。この関数セットのことを**ニューラルネット**（NN）と呼び、特に層の多い（深い）ものを**ディープニューラルネット**（DNN）と呼ぶことが多い。これに最適化技術などの周辺諸々を含む技術領域が**ディープラーニング**となる。

・次式は入力が3つあるニューロンのアフィン変換部分を示す数式である。ベクトル  $x$  が入力に相当し、ベクトル  $w$  が**重み**・ $b$  が**バイアス**と呼ばれるニューロンの挙動を特徴付けるパラメータである。これにシグモイド関数や**ReLU関数**といった非線形関数を適用することで1つのニューロンが形成される。

$$w \cdot x + b = w_1x_1 + w_2x_2 + w_3x_3 + b$$

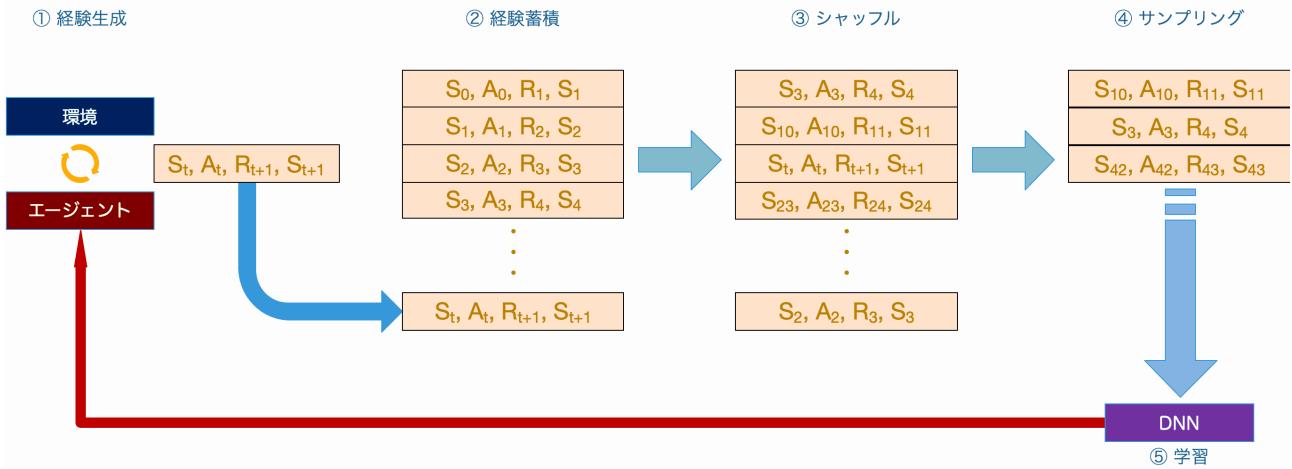
・このニューロンを並列に並べて層状にしたレイヤーを複数重ねることでNN～よく見るディープラーニングの図に相当するもの～が形成される。この各々のニューロンに格納されているパラメータ ( $w, b$ ) を個別に調整することで様々な関数を表現できる。



・近年のディープラーニングはネットワーク構造・活性化関数・最適化手法・その他の構造（ドロップアウト・ゲート）を組み合わせることで多様な領域における性能を達成している。  
・本論文においてはプログラミングフレームワークが充実していて、柔軟性の高い、最適化可能な関数セットだと考えておいて問題無い。

## ■深層強化学習

・強化学習の価値関数あるいは方策関数の関数近似器にDNNを用いたものが**深層強化学習**である。近年の深層強化学習の流行は2015年のDeepMindによる**DQN**（Deep Q Network）がその先駆けである。以下、このDQNで採用されている幾つかの技術について簡単に紹介する。DQNはQをDNNで関数表現したQ学習に、**経験再生**・**ターゲットネットワーク**といった技術を組み合わせたものである。



・**経験再生** (experience replay) とは、行動時に経験を獲得し次第即座に学習する (DNN を更新する) のではなく、①先ずはある程度の回数行動して、②経験を蓄積する。③そして、この蓄積した経験を、得られた順番に関してシャッフルしてランダム化した上で、④その中から幾つかをサンプリングして作成した経験セット（ミニバッチ）に対して、⑤バッチ学習を行う。これを定期的に行う。シャッフルする理由は DNN に学習させる経験セットが時系列的に偏ったものにならないようにすることであり、シャッフルしても問題ない理由は環境が MDP だからである。

・**ターゲットネットワーク** (target network) とは、ベルマン方程式の更新時には次時点の状態行動対における Q 値  $Q(s_{t+1}, a_{t+1})$  を用いるわけだが、これを現在の DNN ではなく一定期間固定された DNN で算出する方法である。詳細は省くが、ベルマン方程式は更新対象となる  $Q(s_t, a_t)$  と更新の目標値になる次時点の  $Q(s_{t+1}, a_{t+1})$  を DNN から算出するが、このとき両者の計算に用いる DNN が同一だと更新対象の更新の影響が目標値にも及んでしまい学習が不安定になってしまう。そこで、目標値を算出する DNN は一定期間固定して定期的に更新するようとするのである。

## ■3.保険料

・本節では保険料算出を扱う。但し、保険商品のタイプとしては **YRT** (1年定期) のみを考えることとする。これは、後述するような単純な需要関数（顧客行動）を前提にするならば、シミュレーション環境下において販売契約から発生する利得（割引即時報酬）は新契約価値（VNB）に一致するし、需要についても販売時点の保険料水準のみを対象にすれば十分だからである<sup>13</sup>。

### ■YRT

・まずは簡単な問題として、**1期間**での YRT の保険料の最適化問題について考える。これは実質的には単なる 1変数（保険料）の最適化問題である。具体的な計算式は以下の通りとした<sup>14</sup>。観測量としてはダミーとして固定値 0 の配列を渡すようにしている。

$$n = \max(0, 10 - 0.1 \times \text{premium})$$

$$R = n \times (\text{premium} - 100 \times q)$$

・n : 販売件数

・premium : 保険料（行動に相当）

・q : 死亡率 (=0.001)

・R : 利益（即時報酬に相当）

・図 1 は、左が件数ベース需要曲線（表記は sales）・右が利益（profit）である。需要は、図 1 左に示すように、保険料（premium）が上がっていくと低下していく。図 1 右の利益は最初は保険料が上がるにつれて上昇していくが、ある程度上昇すると保険料が高すぎて需要が低下してくるため下降していくという放物線を描いている。この問題の解析解は保険料 50 のときの報酬 250 である。

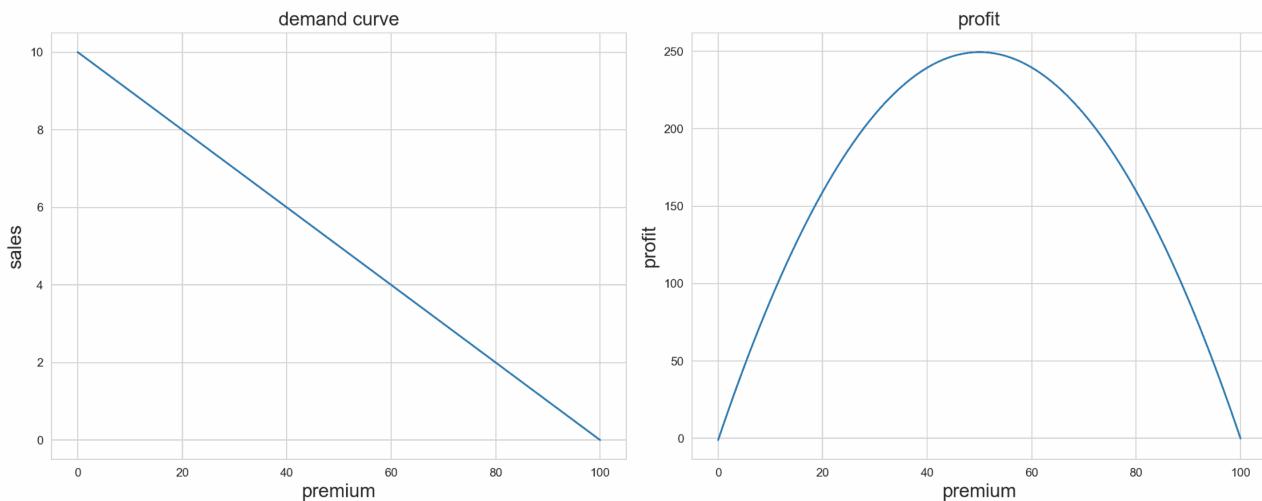


図 1: 需要曲線（左）と利益（右）：X 軸は保険料（premium），Y 軸は各々件数需要（Sales）と利益（profit）である。需要曲線は保険料 0 の場合は 10・100 の場合は 0 であり、保険料の一次関数となっている。利益は保険料と販売

<sup>13</sup> 例えば、新商品の投入や将来的な値下げによる既存商品の解約率の変化、自社の行動による他社の行動の変化、保有の大小による事業費効率や資産運用効率の変化などまで考慮に入れる場合は、自身の将来の行動によって（事後的な）新契約価値が影響を受けることになるのでこれは妥当ではない。しかし、このような長期的な保有情報をベースにすると状態空間が非常に広くなり、計算・実装が困難であるし、保険期間をどうするか、各期間における解約率や転換率をどう考えるかなどパラメータが増えすぎてしまう。特に、後者の問題は実際の数値を使わなければあまり意味のある洞察は得られないものと思われる。但し、これらの影響を完全に無視するとしても技術的には YRT と平準払いは異なっている。というのも平準払いの場合は強化学習エージェントが自力で新契約の価値を推定できるようになる必要があるが、YRT の場合は即時報酬としてそのまま渡されるからである。だから、この YRT への簡略化は新契約価値の計算を学習するプロセスを省略している側面もある。

<sup>14</sup> 強化学習の定式化に従うなら、状態数は 1（あるいは終端状態を含めるならば 2）となる。あるいはバンディット問題と見なす方が自然かもしれない。

件数から決まり、既に式で示したように放物線を描く。この系では保険料を0から上げていくと初めのうちは利益が増加していくが、保険料50を境に値上げによる需要減の方が勝るようになり利益は低下していく。

・図2・図3は2つの方法におけるこの問題の学習結果である。ここでは古典的な**テーブルQ学習**と**ChainerRL<sup>15</sup>**によるDQNによるテストを行った。保険料は本来連続値であるが、両アルゴリズムは離散的な選択肢を前提としているため、保険料は0から100まで1刻みの101の選択肢として扱っている。両者共に概ね30000回程度で収束していることが分かる。

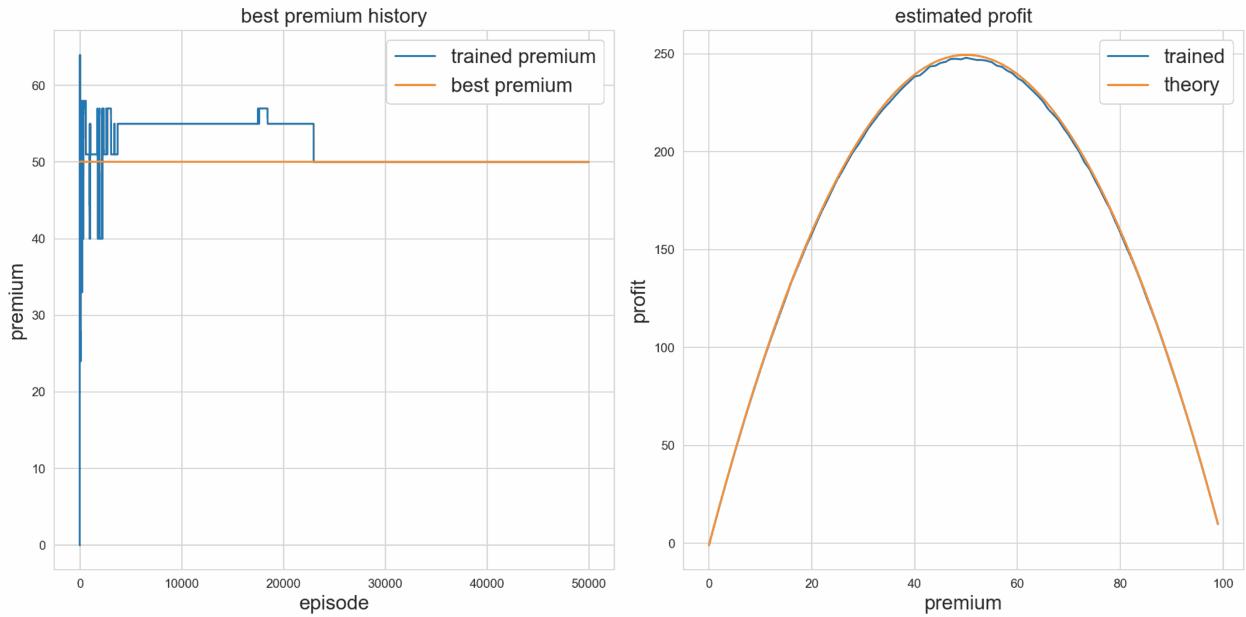


図2: テーブルQ学習における学習曲線（左）と学習されたQ関数（右）：左図においてX軸は学習に要したエピソード数（episode：試行回数）、Y軸は各エージェントの学習しているその時点での最適保険料（trained premium）である。30000エピソード程度で最適保険料（best premium）に収束していることが分かる。右図は学習完了時における各保険料に対するQ値の関係を理論値と学習された値とで比較したものである。X軸は保険料（premium）、Y軸が利益（profit）である。理論値と学習された値がほぼ一致していることが分かる。

<sup>15</sup> <https://chainerrl.readthedocs.io/en/latest/>

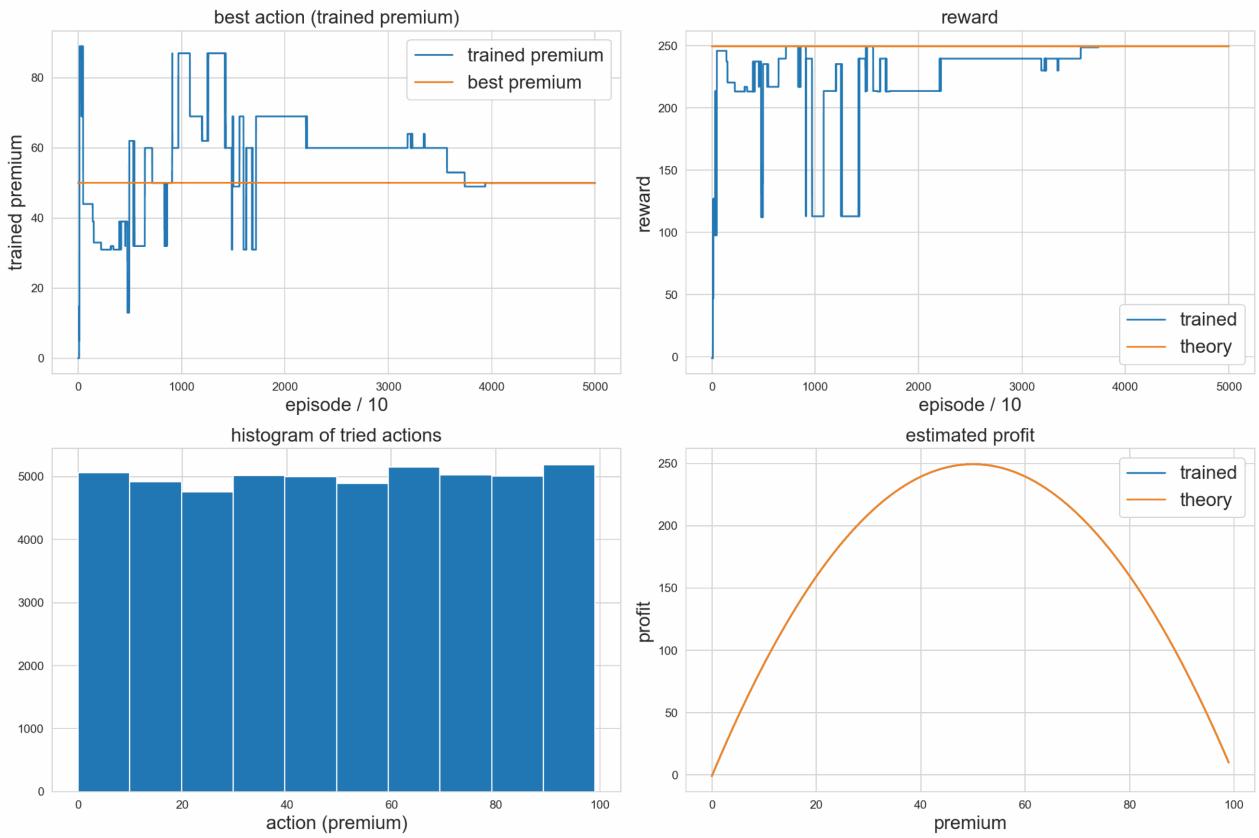


図 3: ChainerRL における DQN の結果：左上は保険料の学習曲線で先程と同様である。右上はエージェントが獲得可能な利益を示しており学習に伴い上昇していることが分かる。左下は学習中にエージェントが試行錯誤した行動のヒストограмであり、満遍なく行動を試していることが分かる。右下は先程と同じく学習された Q 関数を示している。

- ChainerRL の DQN の場合、全部で 50000 ステップ計算しているが、最適値へと収束しているのは 40000 回を超えたあたりであり、かなり時間がかかることが分かる<sup>16</sup>。今回、選択肢（保険料パターン）は既に述べたように 101 個しかない。にもかかわらず概ね 400 倍近い試行が必要であることが分かる。

## ■更新あり YRT

- 次にもう少し強化学習らしい問題として、**更新を考慮した YRT** の保険料の最適化問題について考える。基本的な設定は前項と同じだが、需要関数が保有契約の更新時と新契約とで異なるようになる。具体的には以下の数式でモデルすることで、更新時の価格弾力性が低下するようにした（ $\text{clip}()$ <sup>16</sup> は、括弧内の値を 0~1 にクリップすることを示す）。

$$\text{IF}_0 = 0$$

$$\text{NB}_t = 1 - \text{IF}_t$$

$$\text{NBdemand}_t = \text{clip}\left(1 - \frac{P_t - P_{\min \text{NB}}}{P_{\max \text{NB}} - P_{\min \text{NB}}}\right)_0^1$$

$$\text{IFdemand}_t = \text{clip}\left(1 - \frac{P_t - P_{\min \text{IF}}}{P_{\max \text{IF}} - P_{\min \text{IF}}}\right)_0^1$$

<sup>16</sup> つまり、50000 回のトライ&エラーをしたということである。ハイパーパラメータは学習率に関して幾つか試したが、劇的に早くなるようなことはなかった。

$$IF_{t+1} = NB_t \times NB_{\text{demand}_t} + IF_t \times IF_{\text{demand}_t}$$

- ・IF : 年始保有
- ・NB : 年始市場残余部分
- ・P : 保険料 (行動に相当)
- ・ $P_{\min NB}, P_{\max NB}, P_{\min IF}, P_{\max IF}$  : 保有契約・新契約の価格感応度パラメータ ( $=0, 0.1, 0.1, 0.2$ )

・エージェントが状態から受け取る観測量は(時刻 t, 保有 IF, 資金 A)の組である<sup>17</sup>。行動はその時刻における保険料、即時報酬はその年の利益で与えられる(保険金支払は事業費に含めた)。その期の即時報酬に応じて資金が増えていく。簡単のため、破産は扱わないことにした。以上を数式で表すと以下の通りとなる。

$$A_0 = 0$$

$$R_{t+1} = A_t \times i + \left( IF_{t+1} \times (P_t - E_v) - E_f \right) \times (1 + i)$$

$$A_{t+1} = A_t + R_{t+1}$$

- ・A : 資金
- ・i : 利率 ( $=0\%$ )
- ・ $E_v$  : 変動事業費率 ( $=0$ )
- ・ $E_f$  : 固定事業費 ( $=0$ )

・図 4 左は需要曲線である。新契約 (NB) の需要曲線は青で、既契約 (IF) の需要曲線は緑で示されている。新契約の需要曲線は 0 から 1.0 始まりで低下していき 50 でゼロになるのに対し、既契約の需要曲線は 50 までは一定でそれ以降低下していき 100 でゼロになるようしている。尚、離散型の方法で計算させる場合があるため、図における保険料の値は 100 倍になっている。次も同様である。

・図 4 右は様々な保有における保険料設定下の利息を無視した即時報酬である。保有が小さい場合(グラフで上の方に相当)、保険料をやや左寄り(低め)に設定するのが最大となるが、保有が大きくなってくると保険料を真ん中に設定するのが最大となる。これは既契約は価格弾力性が低いため、より高い保険料を設定可能であることを反映しているものである。このように、保有に応じて即時報酬に対する最適な保険料が異なる上に、現在の保険料に応じて次回以降の保有が影響を受けるモデルとなっている。

---

<sup>17</sup> ここで時刻 t を観測量に含めることは必須である。例えば保有がゼロの場合を考えてみよう。これが初期であれば保険料を下げて保有を確保して次から値上げして利益を取り返すという方策を取ることが可能だが、最終時点ならば取り返すことがもはやできないので、そこそこの保険料で販売量と利益を確保するしかない。つまり、最適な保険料は時刻 t に依存するのである。このため、時刻 t を観測量に含んでいないと安定した学習ができない。これは期間が短い場合に顕著である。

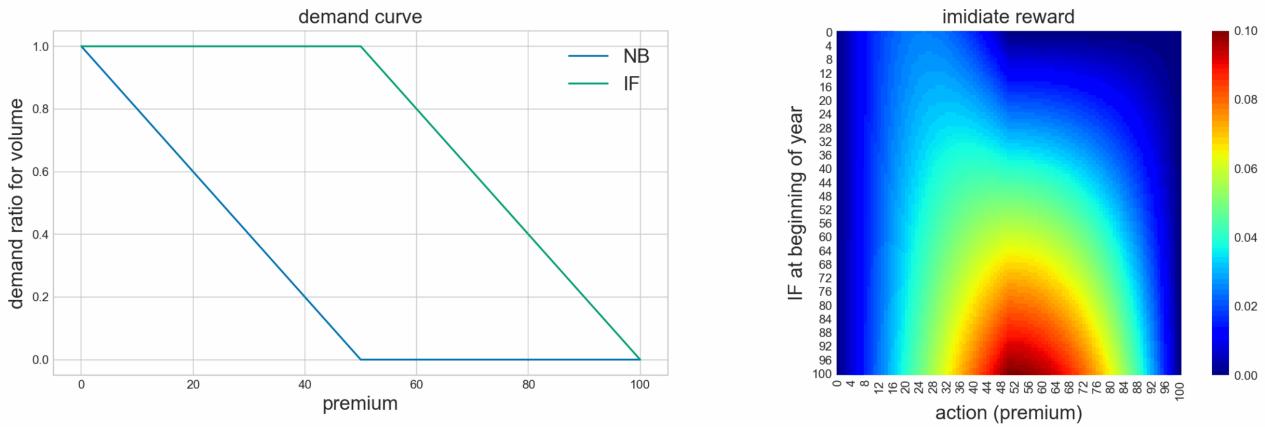


図 4: 需要曲線（左）と即時報酬（右）：今度の需要曲線は既契約の継続（IF）と新契約（NB）とで変化するようにしている。新契約の場合は 0 始まりで低下していき 50 で 0 になるのに対し、既契約の場合は 50 までは一定でそれ以降低下していき 100 で 0 になる。つまり、既契約の方が価格感応度が低い。右側は X 軸が行動（今期設定する保険料）、Y 軸が期初保有（上の方が少ない保有に対応しているので注意）とした場合の今期に獲得される即時報酬を表している。期初保有が小さい場合（グラフで上の方に相当）、即時報酬が最も大きくなるのは左側 1/3 程度の保険料だが、期初保有が大きい場合（グラフで下の方に相当）、即時報酬が最も大きくなるのは真ん中あたりの保険料となる。

・この問題は非常に単純に見えるが、図 5 に示すように、毎回の保険料をランダムに設定した場合の（保有、資金）の分布の時間発展を見ると、 $t=2$  の時点で既にかなり複雑になっていることが分かる。このため、本節では最適解が確定するように既に示したような極端な需要関数にしている。また、性能検証が難しくなるため、ランダムに 10 万回試行した場合との比較を行うことにした。

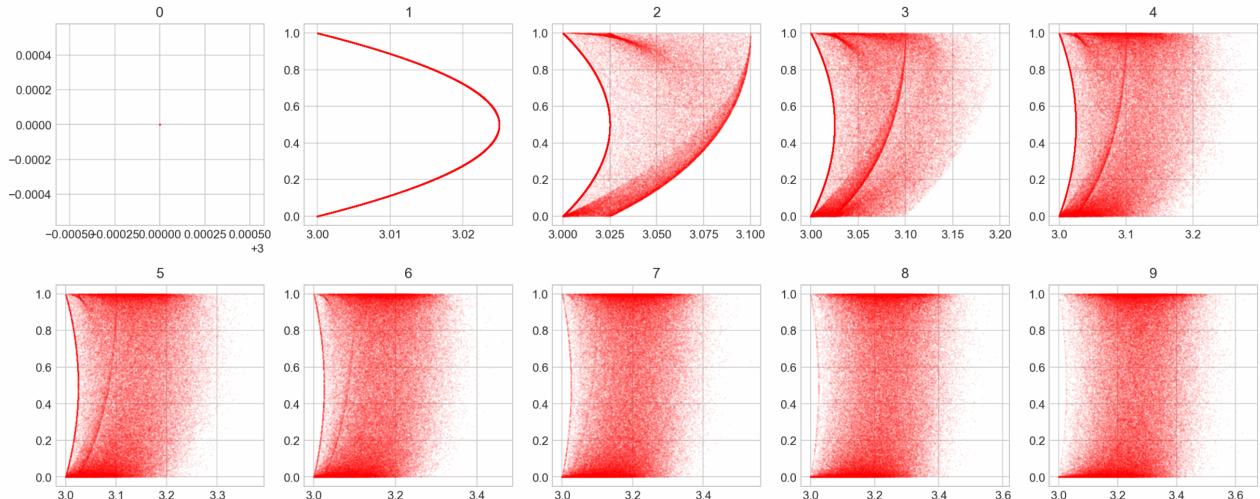
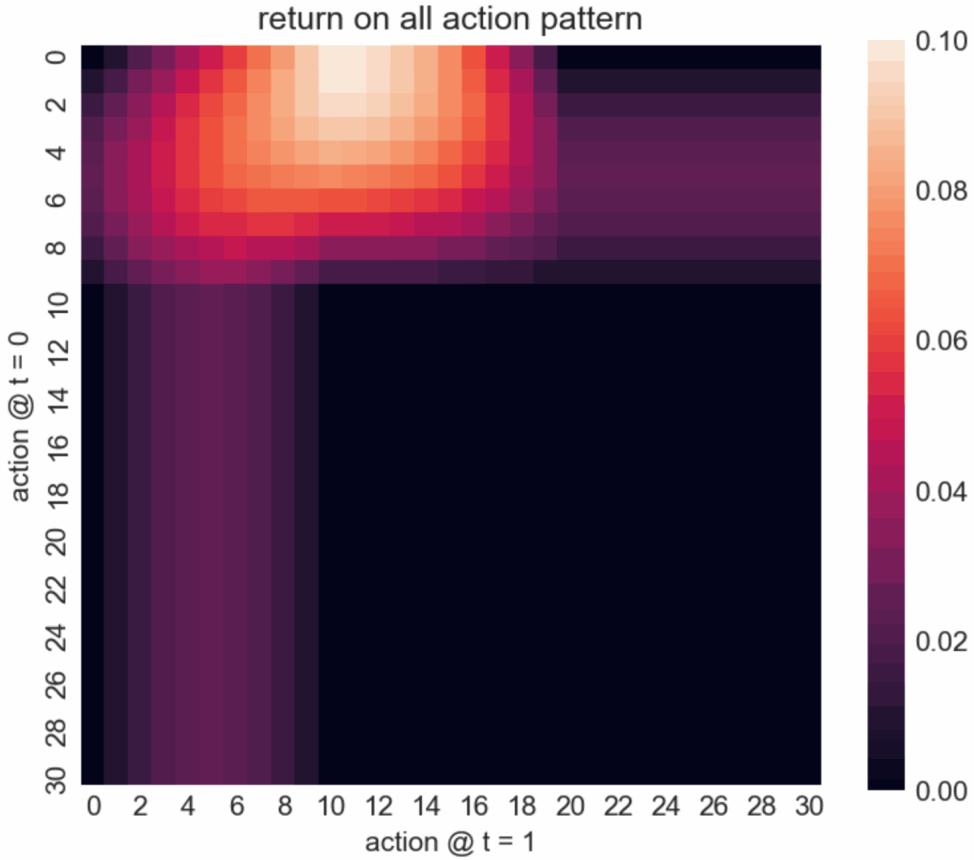


図 5: 保険料を一様分布で発生させた場合の保有と資金の時間発展：各グラフの上部にある数字が各時点を表している。X 軸が資金、Y 軸が保有である。開始時点 ( $t=0$  で左上) は保有も資金も 0 である（中央の 1 点）。 $t=1$  では放物線状になり、 $t=2$  以降は複雑な分布を示すことが分かる。最終時点 ( $t=9$ ) での資金の最大値は 3.6 程度であることが予想される。

## ◇2 期間モデル

・先ず、簡単な例として 2 期間 ( $t=0, 1$  のみでの保険料設定) の例を示す。この場合は行動が 2 時点でしか行われないので、図 6 に示すように全ての行動パターンの結果を 1 つのヒートマップ上に示すことができる。 $t=0$  で 0・ $t=1$  で 10 に設定した場合が最も高くなっていることが分かる。これは、 $t=0$  では保険料を低くして保有を確保した上で、 $t=1$  で値上げするという行動に相當している。



**図 6:** 2期間モデルにおける行動の組合せに対する利得：Y 軸が  $t=0$  における保険料、X 軸が  $t=1$  における保険料である。ヒートマップの色はエピソード全体で獲得できる即時報酬合計～利得～を示している。このグラフから、最適な行動は  $t=0$  で 0,  $t=1$  で 10 であることが分かる。これは、 $t=0$  で保有を稼いで  $t=1$  で値上げして利益を得るという方策になっている。尚、縦と横に線状のパターンが見られるが、これは  $t=0, 1$  のどちらか片方で需要がゼロになるような保険料を設定した場合である。 $t=0$  で 10 以上の保険料を設定した場合、 $t=0$  における即時報酬はゼロとなり終了時の保有もゼロとなるため、最終結果は  $t=1$  で保険料を幾らにしたかのみに依存する。同様に  $t=1$  で保険料を 20 以上にした場合は  $t=1$  での即時報酬はゼロになるため、最終結果は  $t=0$  で保険料を幾らにしたかのみに依存するのである。

#### ◇10期間モデル

- ・上述の設定で 10 期間で学習させた場合の結果は図 7 の通りである。学習した行動は、初回のみ保険料=0 で保有を最大化した後は保険料=10 に設定して利益を維持するようになっている。また、NAF を除くどのアルゴリズムでも 10000 回（グラフ上は  $t=100$ ）程度で学習できていることが分かる。また、ランダムに 10 万回試行した場合（グラフ上は  $t=156$  相当）の最高利得は 0.719 であったため、ランダムよりはかなり高い性能を示していることが分かる。尚、期間を長期化させると学習速度は落ちるもの、30 期間程度までなら学習できることを確認している。

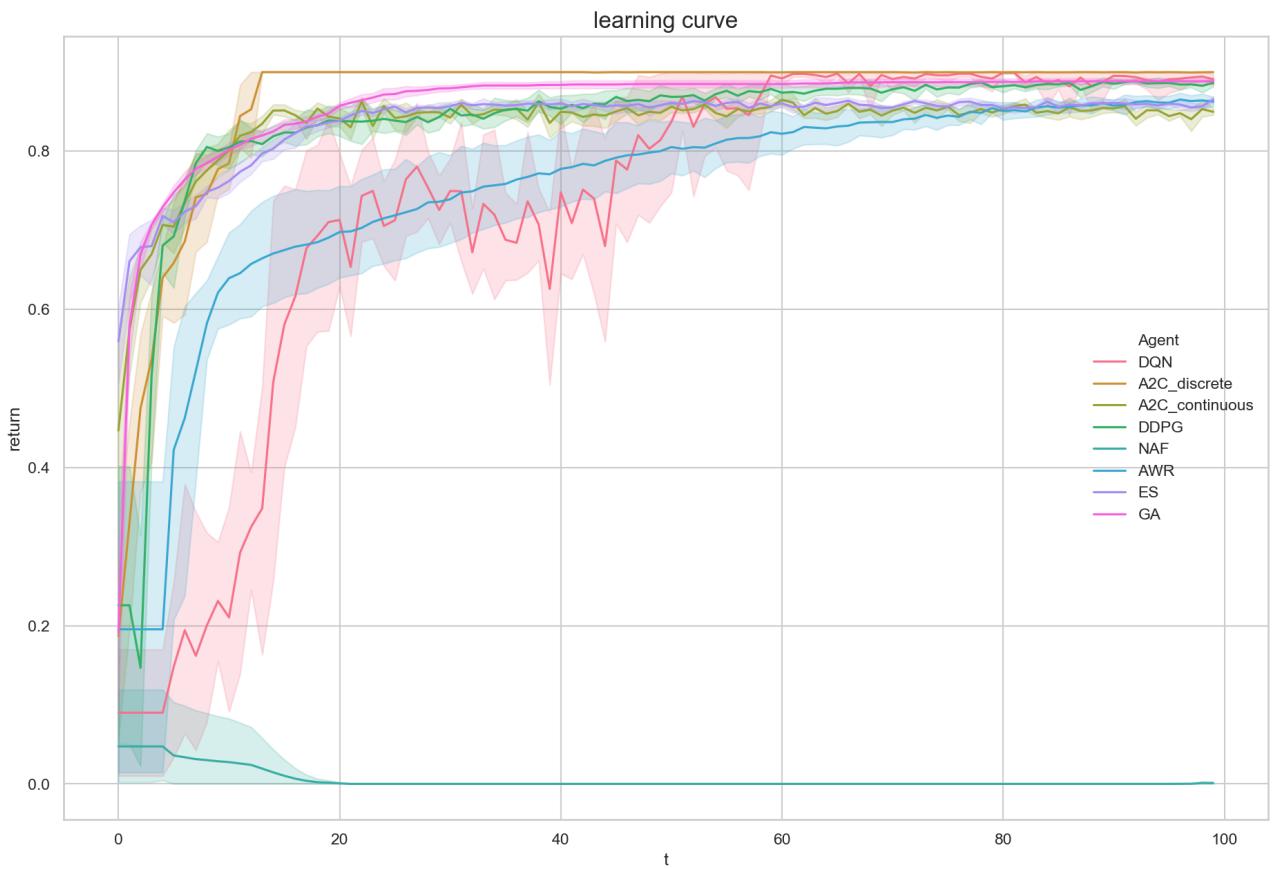


図 7: 10 期間モデルにおける学習曲線：NAF を除くと概ね学習できていることが分かる。連続型のアルゴリズムの最終性能が若干悪いように見える。学習率減衰が必要なのかもしれない。

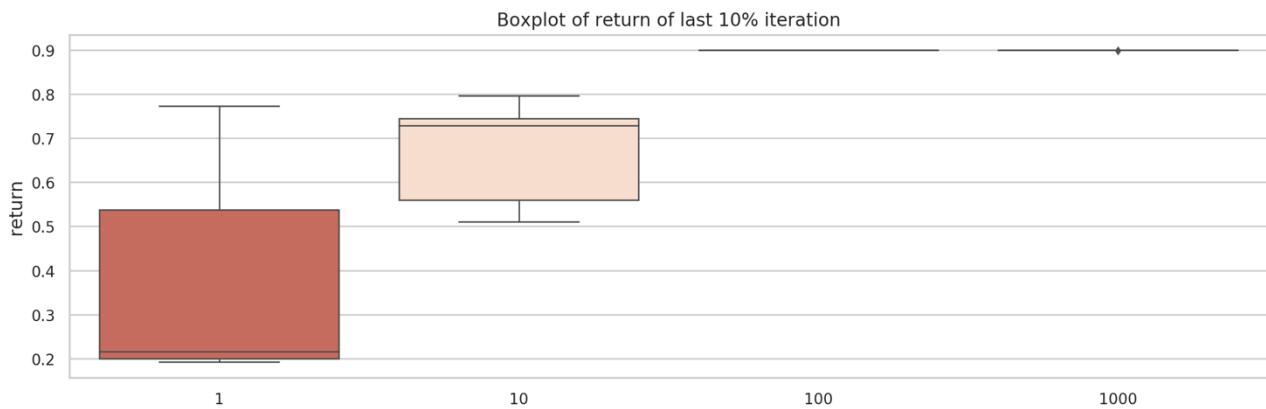
#### ◇環境を介さない学習

・強化学習では通常、環境が必須要素ではあるが、経験再生を用いる方策オフの方法の場合は環境が無くとも経験メモリに蓄積された経験だから学習を行うことが可能である。通常の方法では会社のトランザクションデータなどから死亡率・解約率などの基礎率を算出し、それを元に保険数理などを組み合わせて環境モデルを作成し、それに強化学習を適用するということになるが、この方法では最初の会社のトランザクションデータを経験メモリに直接格納して基礎率の算出や環境モデルの構築を経由せずに直接に方策を形成することになる。勿論、経験メモリに蓄積されていない経験を新たに取得することができないし、汎化に関してはNNに任せるままであるため、通常の方法に比べて性能は劣ると考えられる。しかし、基礎率や環境に関する**専門知識が一切不要**なので、データが十分にあり、かつ外挿の度合いが小さい問題設定ならば有効な代替手段になると考えられる。そこで、最後に本問題でこの方法がどの程度の性能を出せるかを検証する<sup>18</sup>。

・DQNに適用した場合の結果は図8の通りである。横軸は経験メモリに蓄積してあるサンプルサイズ÷64である<sup>19</sup>。サンプルサイズ100（実際には6400）で最高性能に到達していることが分かる。

<sup>18</sup> この方法が有効な問題では、プログラマティックな観点では経験率分析・モデリングがかなりの部分不要になってしまうので、アクチュアリー業務に与える影響は大きいと考えられる。(18)でも示唆されているように、今我々が考えているフレーム・分節での説明可能性というのではなくても必要とは限らない。

<sup>19</sup> コードを使い回すために、完全ランダムでの行動を経験メモリに蓄積し、1度だけその経験メモリからのバッチ学習を100回行うこととした。ターゲットネットワークはミニバッチ学習10回毎にアップデートした。このため、図7に比べると少ないステップ数でもNNの更新回数が多いため性能は高くなっている。



**図 8:** 環境を介さない場合の性能分布 : X 軸は学習に利用したサンプル数÷64 である。1 エピソードは 10 サンプル相当なので、左から概ね 6 回・64 回・640 回・6400 回の試行に相当している。この問題の行動パターンは全部で  $2^{10}$ 通りあるので、ランダムの 6400 回で最適行動にならないことはある意味当たり前ではあるが、DQN エージェントは 640 回分のランダムなデータから最適行動を構築できている。尚、グラフタイトルは last 10%だが、これは単純にコードの関係で特に意味は無く、学習終了時の性能を表している。

- ・先ず、ランダム試行との比較だが、ランダムの場合、それぞれ同等のサンプル量での最高性能は 10 で 0.57, 100 で 0.613, 1000 で 0.633 であった。つまり、ランダムに対して明らかに高い性能を示していることが分かる。
- ・更に、ランダムはエピソード単位で評価しているので完璧なひとつながりの時系列の情報があるのに対し、経験メモリに蓄積されているものは時系列がシャッフルされた断片的なデータであるということも重要だと思われる。つまり、断片的なデータを繋ぎ合わせて最適行動を作り出せているのである。実用上はエピソード単位のまとまったデータよりは部分部分の断片的なデータの方が多く集めやすいため、この方法の適用可能領域は大きいと考えられる。

## ■対戦型 YRT

・更に強化学習らしい問題として、前項の問題を複数のプレイヤーでプレイする問題について考えた。マルチエージェント強化学習ではなく、単純な自己対戦形式での評価を行った。方策のベースとなる NN 自体は両プレイヤー共通とし、より多く利益を得ることを学習させることにした。状態認識として、各プレイヤーは自分と相手の観測量、即ち保有・資金が分かっているものとした。この上で、無保険者・保険会社 A・保険会社 B 間の遷移行列を保険料に基づくソフトマックス関数により定義し、同一区分に留まる場合にはスイッチングコスト分だけ保険料が安く扱われるようにして継続に傾くように調整した。更に、資金が負になった場合は破産したと見なしして行動停止状態になるようにした。

・結果としては残念ながら遷移行列に対して非常に不安定である上に結果も安定しなかった。しばしば開始直後に保険料ゼロに設定してその後一定という行動を学習することはあったものの、やはり安定しなかった。学習済のエージェントをランダムエージェントと対戦させても成績は振るわないどころか若干弱いという結果ですらあった。

・単純に不安定なのだとすれば、現実に近い需要曲線と結果の変動幅の評価がセットでなければ意味は無いし、何らかの数理的な問題があるならば別途検証しなければならないが、今回はそこまでの時間は無かった。ただ、元々この分野は難しく、数理的（特にゲーム理論）にも考えなければならないことが多いようなので、もう少し先行研究について調査した上で再トライしたい。

## ■まとめ

- ・単純な YRT モデルにおいて保険料の最適化が行えることを確認した。
- ・方策オフの場合、環境そのものが無くともそれなりの方策を形成できることを確認した。
- ・対戦モデルは基礎的な検証が必要。

## ■4.資産運用

- ・本節では資産運用の簡単な問題として資産配分問題・デュレーションマッチングを扱う。いずれも比較的安定して学習可能だが、資産配分問題に対しては学習効率が非常に悪いことが示された。

### ■資産配分問題

- ・先ずは古典的な資産配分問題を扱う。幾つかの投資対象資産があり、それらのリターン（利益率の期待値）とリスク（標準偏差）のみが得られており、資産を上手く組み合わせることで期待リターンを上げつつリスクを下げる問題である。リターン・リスク選好に関しては様々に考えられるが、ここでは簡単のため、両者の選好スケールは同じであるとした。つまり、即時報酬は単純にリターン - リスクとした。以上を数式でまとめる以下の通りとなる（ $\mu$  が期待リターン・ $\sigma$  がリスク・ $r$  が資産配分を示すベクトルである。尚、簡単のため全ての資産のリターンは独立であるとした）。

$$R = \sum_i r_i \mu_i - \sqrt{\sum_i r_i^2 \sigma_i^2}$$

- ・資産配分を考える上では本来はその合計値が 1 になっていなければならないが、ここでは簡単のため、行動  $a$  としては各資産について 0~1 の間で決定した上でそれを正規化することにした。離散型のアルゴリズムでは各資産に対して 0 から 1 までの 0.1 ずつの 11 の選択肢を設定した上で同様に処理した（例えば 2 資産の場合は  $11 \times 11 = 121$  通りの選択肢となる）。

$$r_i = \frac{a_i}{\sum_j a_j} \quad 0 \leq a_i \leq 1$$

- ・観測量としては、各資産のリターン・リスクを与えていた。
- ・この問題は前節の YRT の最初の問題と同様に 1 時点のみであり静的である。但し、YRT では決定対象はただ 1 つの変数～保険料であったのに対して、こちらは複数の変数～資産配分が決定対象となる。

### ◇2 資産モデル

- ・先ずは 2 資産の場合を考える。リスク・リターンは各々 (0.01, 0.01) ・ (0.05, 0.05) とした。この場合に選択可能な資産配分で達成可能なリスク・リターン関係を示したものが図 9 である。上段の X 軸がリスク（標準偏差 : sigma）・Y 軸がリターンであり、可能なポートフォリオのリスク-リターン分布を表している。下段左がランダムに配分した場合の目的関数～この場合はリターン - リスクへの分布を示している。下段右は様々な行動における目的関数の値を示したヒートマップである。既に述べたように 2 つの資産の配分は一旦 0 から 1 の間で独立に決定され、正規化された上で環境に入力されるが、ここではその正規化前の状態で表現している。これを見ると、やや右下、即ち資産 0 を多目にした方が良いことが分かる。

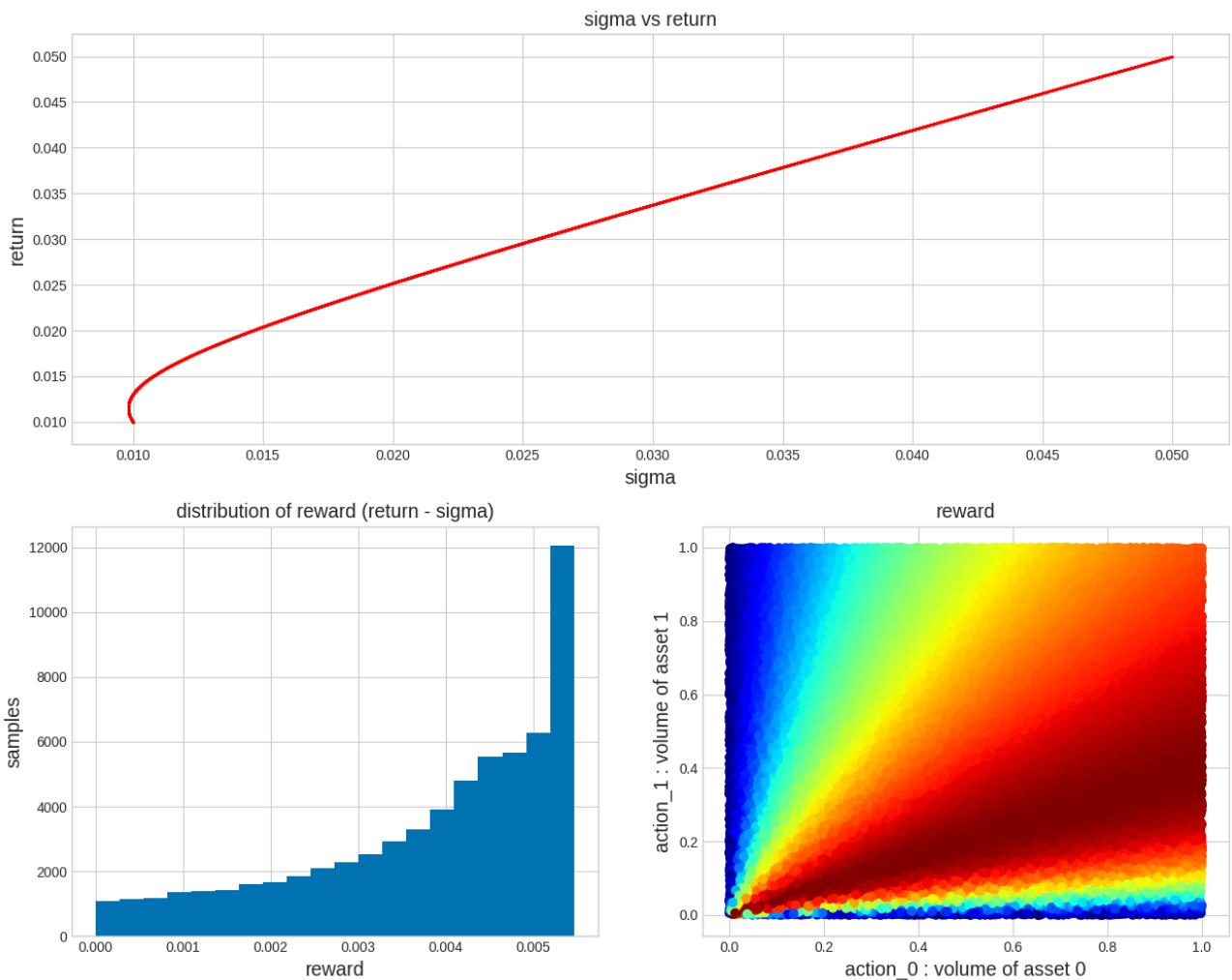


図 9: 2 資産モデルのリスク-リターン関係（上）, ランダム行動の際の目的関数の分布（左下）, ランダム行動の際の目的関数（右下）：リスク・リターンが(0.01, 0.01), (0.05, 0.05)である 2 つの資産における配分問題を考えた際の各種の値である。

・学習させた結果は図 10 の通りである。どのアルゴリズムも概ね 10000 回（グラフ上は 100）で収束していることが分かる<sup>20</sup>。尚、各資産のリスクとリターンは固定している。つまり、この問題を解いた強化学習エージェントは他の資産配分問題には使うことができない<sup>21</sup>。そのようなケースは最後に扱うこととする。

<sup>20</sup> DQN は  $11 \times 11 = 121$  個の点からしか資産配分を選べないが、それでも十分なスコアを達成することができる。

<sup>21</sup> いつも同じ値が出てくる、というわけではないことに注意。NN の推論自体は通常通り行われるので何かしらの値は出てくるが、全く意味をなさないということである。

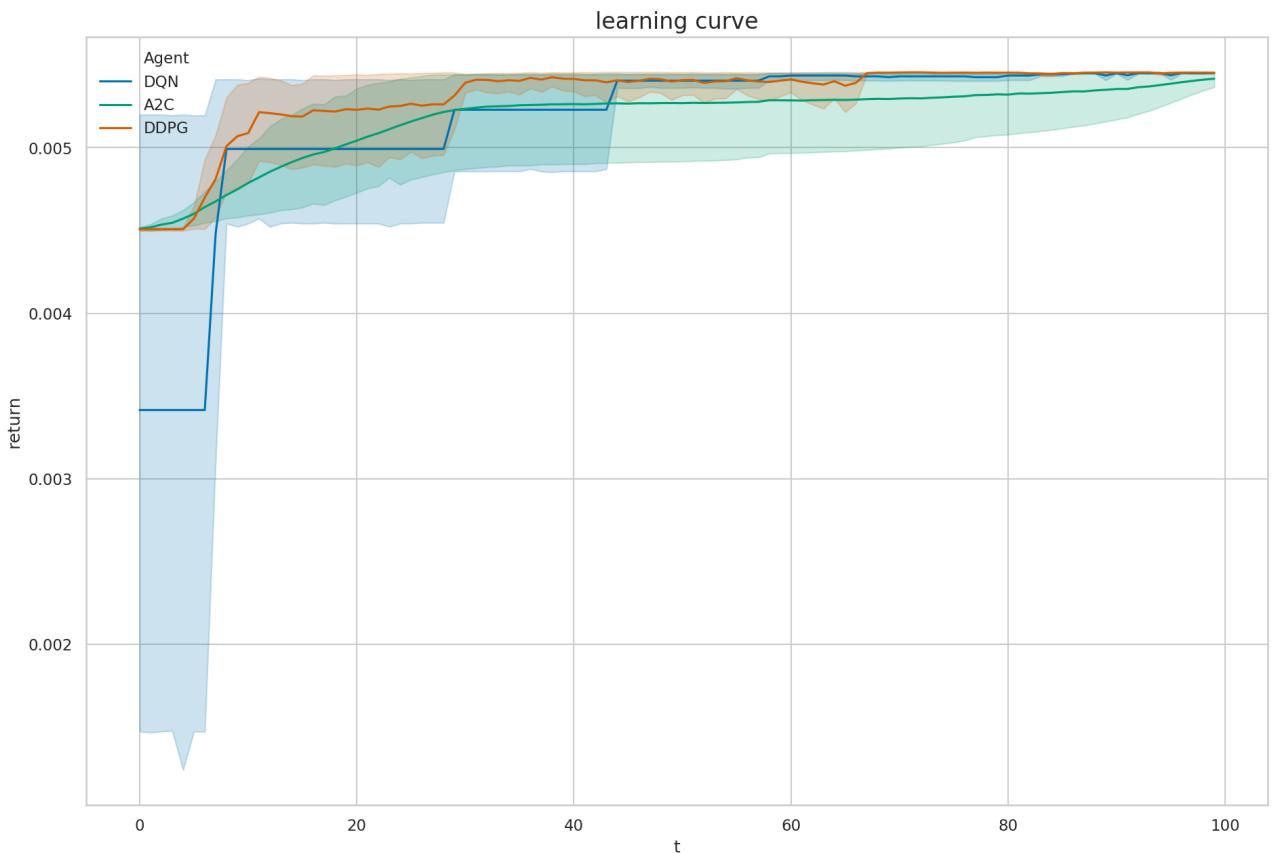


図 10: 2 資産の場合の学習曲線：どのアルゴリズムも 10000 回（グラフ上は 100）で概ね収束していることが分かる。A2C は学習が若干遅いものの安定しており緩やかである。DQN はステップ状に改善が生じている。

## ◇5 資産モデル

・ 続いて 5 資産の場合について考える。ここでは図 11 左に示すようなリスク・リターン特性を持つ 5 資産を考えることにした（ハイリスク・ハイリターン）。図 11 の右側は 2 資産の場合と同様だが、5 資産なので 2 資産の場合のようなヒートマップは作れない。代わりに、リスク・リターンを見ると、先程の放物線状とは異なり面状になっていることが分かる（このグラフそのものはランダム行動から生成しているため、密度分布のようになっているが、実際には明確な境界線がある）。

・ 学習させた結果は図 12 の通りである。DQN は性能のみならず計算時間が劇的に悪化し、実験環境においては 2 資産の場合に 1 分程度であったところ 60 倍程度の 60 分強となった。これは、選択肢の数が  $11^5 \sim 10$  万程度になり、NN のパラメータ数が  $32 \times 11^2 \sim$  のオーダーから  $32 \times 11^5 \sim$  のオーダーになるため計算負荷が非常に大きくなつたためと考えられる。また、状態数が 1 つであるため NN の汎化も機能しておらず、全ての選択肢に対して単純に試行錯誤で経験を蓄積しているのみなので学習効率が非常に悪い。計算ステップ数は 10000 としているためサンプルサイズは 64 万程度だが、選択肢の数は 10 万程度なので、概ね選択肢数の 6.4 倍のサンプルを使用していることになる。これは、2 資産モデルにおいて選択肢数 100 程度に対してサンプルサイズが 64 万で 6400 倍程度だったのと比較すれば圧倒的に少ない。このように、理論的に学習可能かどうかはさておき、資産の種類が増えると離散型のアルゴリズムで対処するのは現実的に不可能になってくる（所謂次元の呪い）。

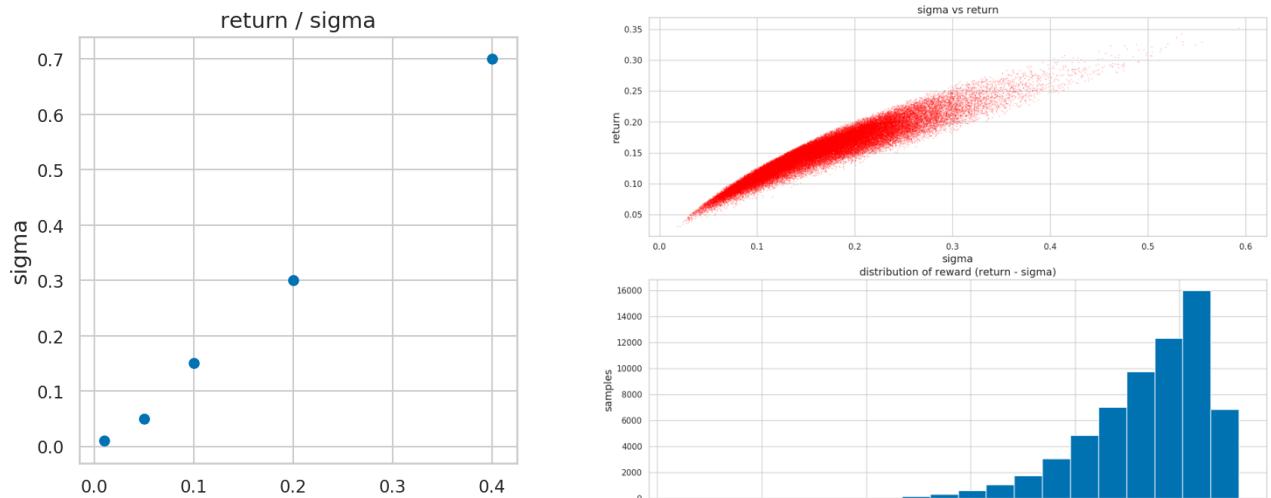


図 11: 5 資産の場合の各資産のリスク-リターン関係（左），ランダムポートフォリオの際のリスク-リターン関係（右上），ランダム行動の際の目的関数の分布（右下）：ランダムポートフォリオの際の分布が先程と異なり曲線から密度分布になっていることが分かる。

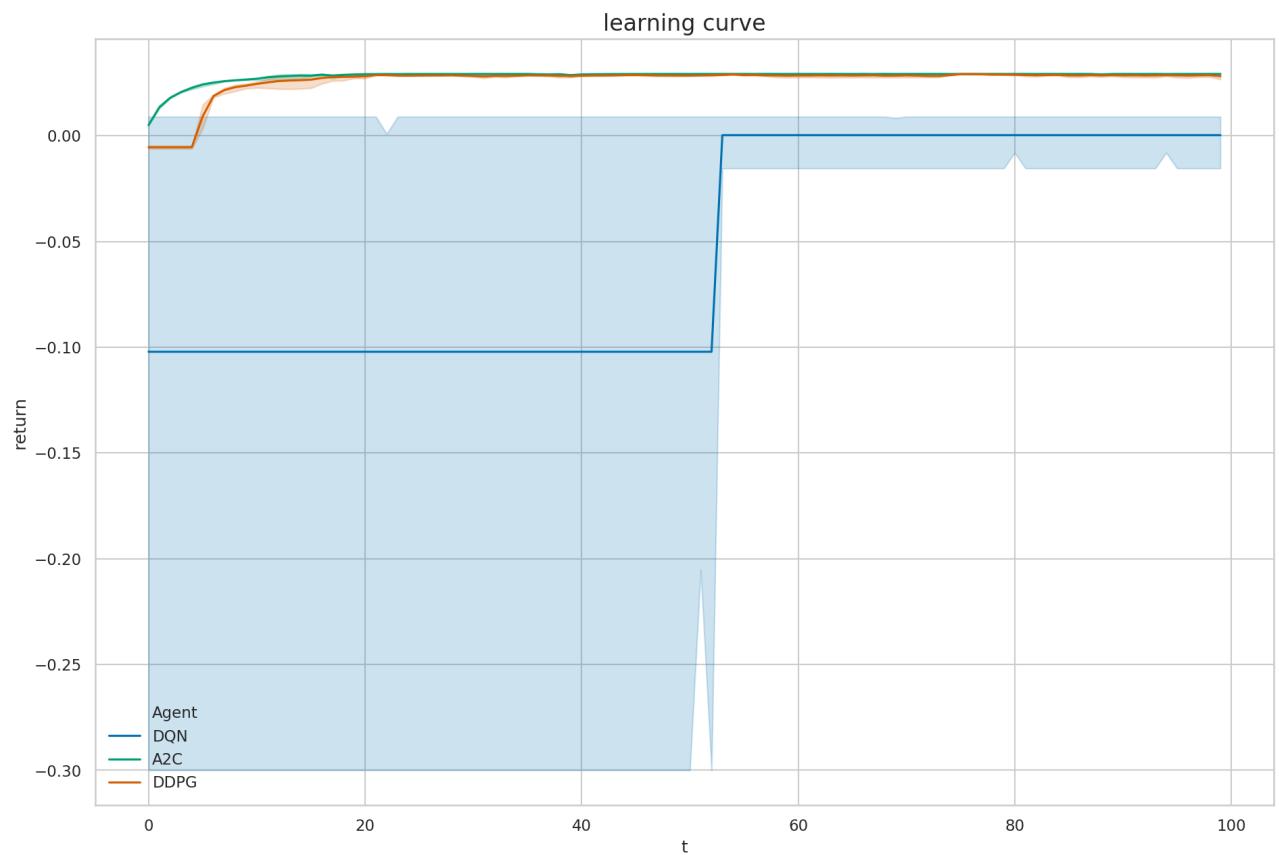


図 12: 5 資産の場合の学習曲線：A2C と DDPG は安定して収束しているが，DQN は収束しておらず，更に最も上手くいったケースでも他 2 つに比べて性能が低いことが分かる。

## ◇2 資産汎用モデル

- 最後に，**様々なリスク・リターン環境**下で機能するように学習させる。これには，リスク・リターンをランダムに変えた問題を大量に出して学習させれば良いと考えられる。先程までは 1 つのリスク・リターンの問題のみを学習させたが，様々な問題を学習させるということである。具体的にはリスクが  $N(0.001, 0.1)$  に，リターンが正規分布  $N(0.01, 0.05)$  に従うとして，毎回この乱数からサンプリングしたリスク・リターンからなる資産配分問題を学習させた。

・性能評価も同様に、様々なリスク・リターンの資産群を与えた場合にどのような配賦を行うかで評価した。図 13 右の Q-Q プロットにおいて X 軸は汎用ソルバー<sup>22</sup>による解、Y 軸は同条件下での各々のアルゴリズムの出力である。様々なケースにおいて、ソルバーと同等程度の行動を生成できていることが分かる。但し、莫大な計算時間がかかるため、直接代わりにはならない。

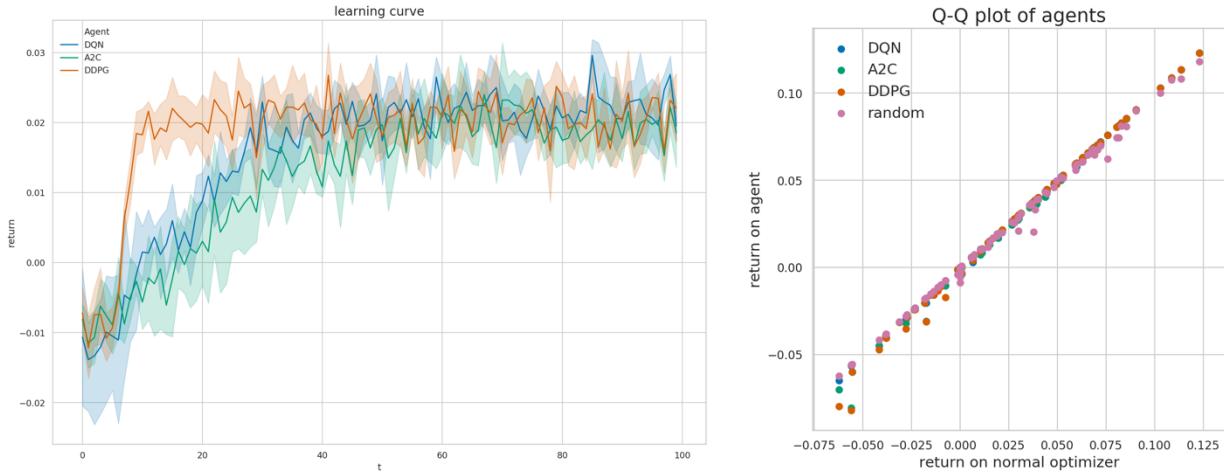


図 13: 様々なリスク・リターン状況における学習曲線（左）と学習後性能の Q-Q プロット（右）：学習曲線（左）上は DDPG が比較的早いものの、最終的にはどのアルゴリズムも同程度に落ちているように見える。Q-Q プロット（右）は X 軸が汎用ソルバーの解・Y 軸が書くアルゴリズムあるいはランダムで 64 パターン生成した場合の最適解を示している。（ランダムを含む）どのケースも概ね 45 度ライン上に乗っていることが分かる。

## ■デュレーションマッチング

・次に、単純なデュレーションマッチングを扱う。10 年目末にキャッシュアウト 10 があり、現在そのキャッシュアウトの現在価値に等しい額の現金を持っているとする。ここで 1 年から 10 年までの年限の債券を毎年始にどれか 1 つに限り購入して毎年末に精算するということを繰り返す。つまり、選択肢はどの年限の債権を買うか、ということとなる。

・即時報酬は価格変動損益の絶対値を負報酬とした。つまり、最適行動は毎回この負の即時報酬をゼロにするようにすれば良い。このため、最高利得はゼロとなる。一旦最適行動から外れてしまうと必ずしも問題は単純ではなくなってくる。以上を数式でまとめると以下の通りとなる（年始・年末・満期の調整は別途行っている）。

$$\begin{aligned} \text{cash}_0 &= A \times \frac{1}{(1 + \text{yield}(0, m - 0))^{m-0}} \\ \text{cash}_{t+1} &= \text{cash}_t \times \frac{(1 + \text{yield}(t, a_t))^{a_t}}{(1 + \text{yield}(t + 1, a_t - 1))^{a_t - 1}} \\ \text{PVCF}_t &= A \times \frac{1}{(1 + \text{yield}(t, m - t))^{m-t}} \\ \text{NetAsset}_t &= \text{cash}_t - \text{PVCF}_t \\ R_{t+1} &= -|\text{NetAsset}_{t+1} - \text{NetAsset}_t| \end{aligned}$$

- ・ cash : 年始資金
- ・ yield(t, a) : 時点 t における年限 a の債権のイールド

<sup>22</sup> SciPy.optimize.fmin を用いた。

- ・ A : 満期キャッシュアウト (=10)
- ・ m : 満期 (=10)
- ・  $a_t$  : 時点  $t$  において購入した債券の年限
- ・ PVCF : 負債キャッシュフロー現在価値
- ・ NetAsset : 純資産
- ・ R : 即時報酬

### ◇満期支払モデル

- ・先ず、最も単純なモデルとして、10年目末に支払キャッシュフローが10発生する場合を考える。この場合の最適解は、1年目には10年ものの債券を買い、2年目には9年もの…10年目には1年ものを買うことである。この場合、毎年の即時報酬はゼロになり、利得もゼロとなる。イールドカーブは実装の都合上、1年債が1%・10年債が10%になるように単純に直線状にした。
- ・結果は図14の通りである。どのアルゴリズムでも安定して学習できているが、やはり最終性能には差があり、特にDDPG・GAを除くと連続系のアルゴリズムは、最高性能には至っていない。

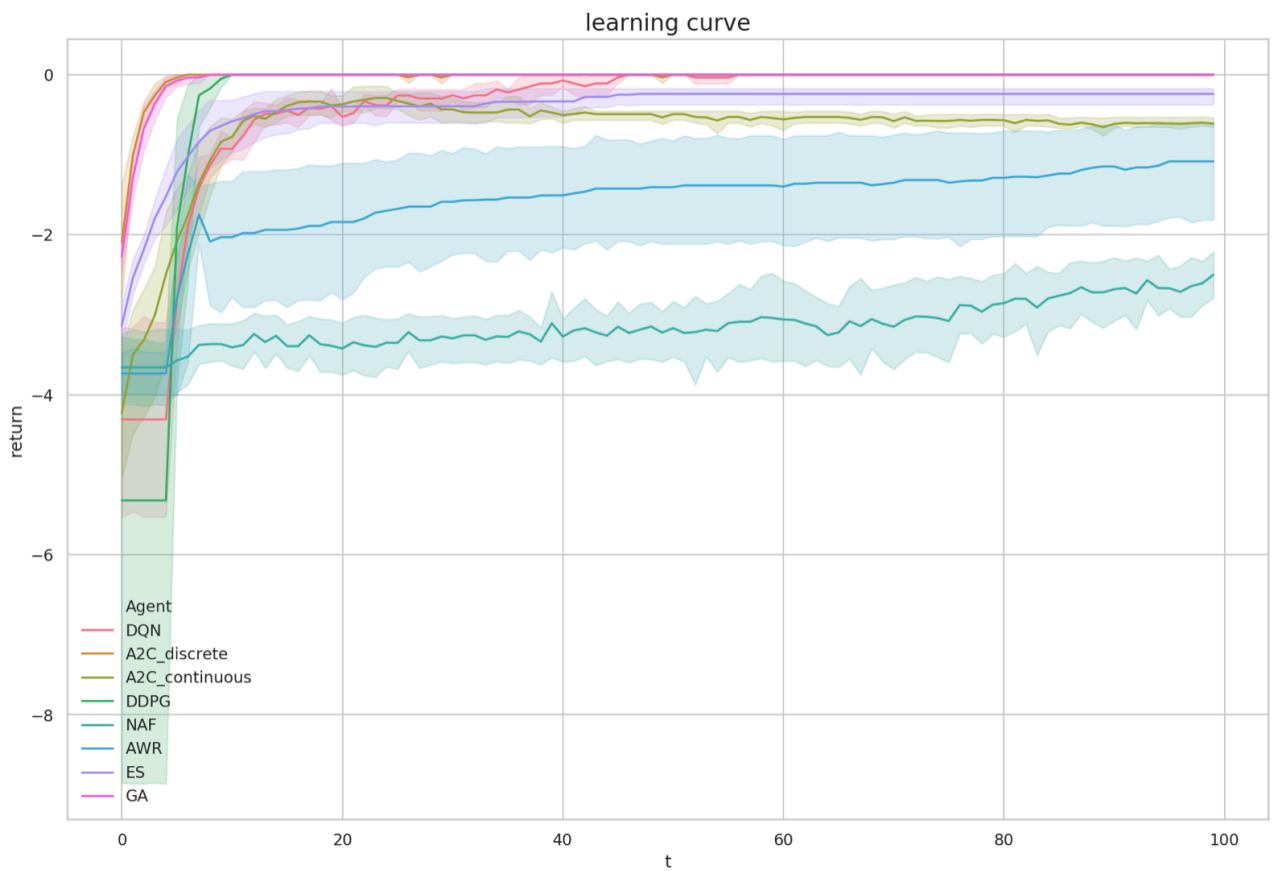


図14: 満期支払モデルにおける学習曲線：NAF・AWR を除けば概ね学習しているように見える。DDPG・GA を除くと連続系のアルゴリズムの最終性能があまり良くない。

### ◇ノイズ環境

- ・上ではイールドカーブを固定して学習を行ったが、本来この問題では残存期間に一致する債券を買えば良いだけなので、イールドカーブが毎年ランダムに変動したとしても最適解は変わらない。そこで、イールドカーブにノイズを加えたときに学習がどのように影響を受けるかを検証した。ノイズは元のイールドカーブに正規分布  $N(1, 0.3)$  をかけることとした。結果として発生するイールドカーブは図15のようになる。

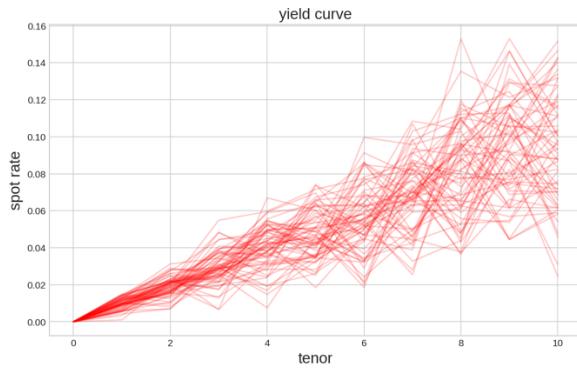


図 15: ノイズを加えた場合のイールドカーブ：長い年限のものほどノイズによる変動が大きい。

- 結果は図 16 の通りである。アルゴリズムごとの最終性能の違いがかなり大きいことが分かる。これは単純にノイズに弱いのか、ハイパーパラメータのチューニングで対応可能なのかまでは検証できていない。但し、DQN と A2C 離散型が最高性能 (0) に到達しているため、連続型のアルゴリズム（残り全て）がノイズに弱い可能性は十分に考えられる。



図 16: ノイズ環境での学習曲線：アルゴリズムごとに最終性能に差が見られる。

#### ◇他のキャッシュフロー

- もう少し複雑なキャッシュフローとして 3 年目と 10 年目のみに 10 のキャッシュアウトがあるものを考える（イールドカーブの条件は最初に戻した）。本問題では投資できる債券は 1 つだけなので、この問題の場合、これまでのように完全なデュレーションマッチングを行うことはできないが、概ねデュレーションマッチングを行うことで損益を安定化させることは可能である。
- この場合も図 17 に示すとおり比較的安定して学習することができる。NAF はともかく AWR の性能があまり良くないが、原因は検証できていない。

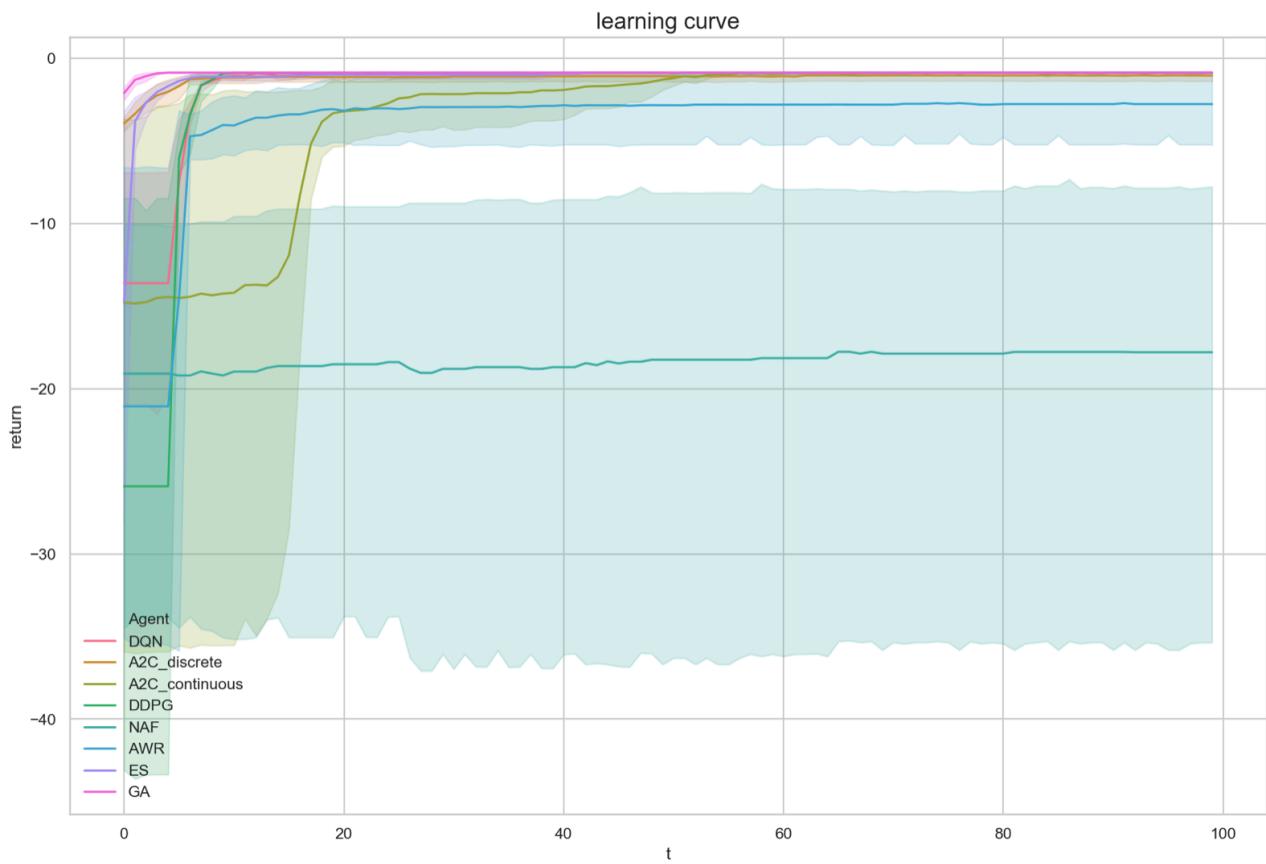


図 17: 別のキャッシュフローにおける学習曲線：AWR の性能があまり良くない。

- ・今回は試していないが、モデルにトランザクションコストや流動性を反映したり、投資方針やリスクアペタイト等を反映した目的に応じた選択肢や即時報酬を設計してキャッシュフローマッチングを行ったりすることも可能だと思われる。

## ■まとめ

- ・基礎的な資産配分問題・デュレーションマッチングについて学習可能なことを確認した。
- ・多資産配分など行動の次元が大きい場合は、離散型の手法を用いると NN のサイズが肥大化して計算困難になる。

## ■5.責任準備金

・本節では責任準備金を扱う。但し、簡単のため **10年一時払年金**のみを考え、更に、**死亡率・解約率・事業費**についてはゼロとする。このように異常に単純化した理由は、この程度の問題でも強化学習の標準的な手法では安定しては解けなかつたため、原因について解析する必要があったからである。また、後述するが、本節では**配当額の決定を通して責任準備金を定義**している。

### ◇問題設定

・責任準備金の計算はアクチュアリーの主要業務ではあるが、計算そのものは強化学習には直結しない。強化学習の問題として定式化するためには、責任準備金の計算に結びつく即時報酬を定義しなければならない。

・先ず「責任準備金の学習」の内容として少なくとも2つが考えられる。1つは所定の責任準備金（例えば標準責任準備金）を学習させること。もう1つは責任準備金そのものを開発させることである<sup>23</sup>。前者は規定の責任準備金を導出できたかどうかに応じて、差分の絶対値に負号を付けた即時報酬を設定すれば良い。しかし、これは単純な教師あり学習に過ぎない。後者は魅力的であり本節でもこのスタンスを取るが責任準備金についてもう少し考えなければならない。

・責任準備金は結局のところソルベンシー（支払能力）を確保するために積み立てる（配当に制限をかける）ものであるから<sup>24</sup>、**十分性**はナイーブには全ての保険金支払ができたかどうかで事後に決定される。しかし、当然ながら「結果的に保険金支払ができたこと」と「ソルベンシーが期中確保されていたこと」は一致しない。アクチュアリーコミュニティが十分と判断できるソルベンシーが確保されていても運悪く破綻することはある得るし、逆に不十分でも運良く破綻しないことはあり得る。あるいは現在リスクとして想定していない事象により破綻するかもしれない。

・しかし、経験のみから最適行動を形成するという強化学習の論理で考える限り<sup>25</sup>、事後に破産したかどうかを基準に据えるのが最も自然であると思われる。保険金支払いに成功したかどうかは評価性のものではなく、その時点の状態から保険数理を用いずに曖昧さなく決定可能<sup>26</sup>であることも、責任準備金そのものを（自動的に）開発させるという趣旨に沿うものと思われる。

・環境モデルが確率的であっても何度も試行を繰り返すことで偶然によるブレは一定程度排除されるだろうし、想定外のリスクについてはそもそもモデルされていないのが問題だと考えられる。

・故に、本節では**全ての保険金支払いができたかどうかだけでソルベンシーの十分性を事後に判定する立場**を取ることにする。そして、安全割増などは環境モデルの確率モデル化・報酬体系の調整などを通じて入れ込むことにする（だから、ここで述べる責任準備金とは営業保険料式に近いものである）。

・一方、十分性だけからは**妥当性**は確保できず、多くの場合に積み過ぎになる。そこで、責任準備金を超えた資産は全て配当として流出させることにし、この配当を即時報酬とすることにする。割引率が設定されていれば早めに配当として流出させる行動の価値が高くなる。つまり、常に資産=責任準備金となる。

・これにより、**全ての保険金支払いに成功しつつ、常に必要最小限の責任準備金を積む行動が最適解となる**ように問題が設定される。つまり、本節では責任準備金と銘打ってはいるが、実質的には配当額の決定問題を扱っている。

### ◇問題の技術的定義

・以上を問題の定義として技術的に簡潔にまとめると以下の通りとなる。

<sup>23</sup> 他には、所定の責任準備金をベースに補正を加えるような方法も考えられる。例えば、法定責任準備金の積み増しは可能だが取り崩しは不可能なように学習させるのは実用上の価値が高いかもしれない。

<sup>24</sup> 勿論、各種の分析に便利であるとか、保守的な現在価値として便利であるとか、契約者持分であるとかといった側面もある。しかし、ここでは単にソルベンシー確保という観点のみから考えることにする。

<sup>25</sup> 全ての強化学習がそうだというわけではないが、本論文では標準的なものを対象にしているのだった。

<sup>26</sup> 将来の見通し・確率などに基づく評価性の会計項目などを用いていない、という意味である。

- ・毎年責任準備金を決定するが、責任準備金以上の資産に関しては全て配当として流出するものとする。そして、この配当を即時報酬とする。利得（即時報酬の現在価値）はそのまま配当利益の現在価値であるから、潜在価値（EV）となる。
- ・責任準備金が積めているかどうかは全ての保険金の支払いに成功するかどうかで事後的に判定する。保険金が支払えない場合、破産したと見なしてペナルティを発生させ、エピソードは終了する。モデルの確率的側面とペナルティと割引率のバランスによって安全割増が決定される。
- ・現在の資産額以上の責任準備金を設定した場合は、現状の資産額と同一の責任準備金を設定したものと見なす。これは、現状の資産額以上かどうかは現状の状態（資産額）から確定可能であることから問題はない（極論、現金のみを資産として扱えば資産側の評価側面も減らせる）。
- ・保険金額は10年目に10発生するのみとし、利率は1%とした。死亡・解約・事業費は全てゼロ、ハードルレートは5%，初期資産は15とした。破産ペナルティは-50である。
- ・この問題には大域解と局所解がある。大域解は正しい責任準備金を常にピッタリで積み続けるものである。局所解は開始直後に資産全額を配当として流出させ10年目に破産ペナルティを受けるものである<sup>27</sup>。すぐ後に述べるが、これにより問題は非常に難しくなっている。これはイメージとしては図18のようになる。

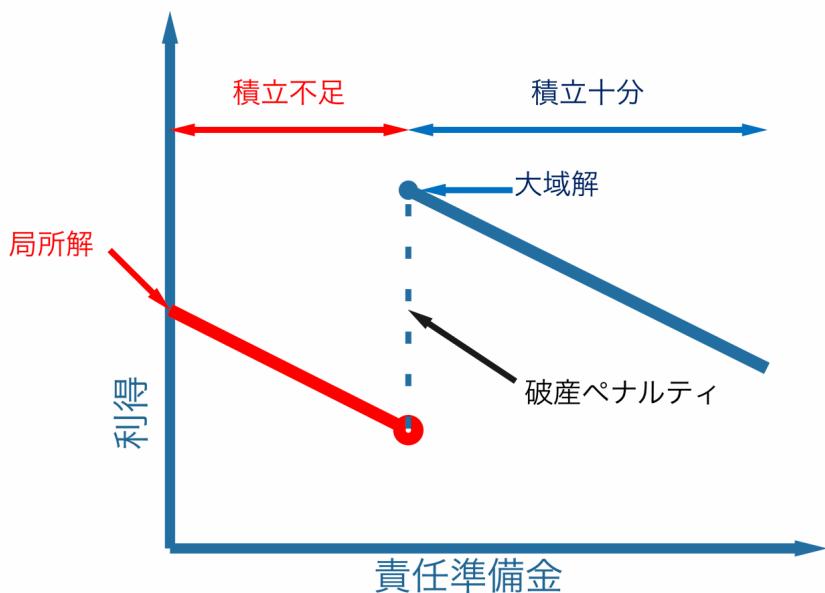


図18: 責任準備金レベルと利得とのイメージ：右から責任準備金を減らしていくと、ハードルレートの効果で利得は上昇していくが、一定値を切ると積立不足になり破産するため、破産ペナルティを受けて利得は不連続的に下落する。この問題の大域解は丁度いい責任準備金を積み続けることだが、局所解として責任準備金を最低レベルにする～即ち開始直後に全額配当として流出させるというものがある。

#### ◇安全割増

- ・ところで、支払債務のキャッシュフローの現在価値の期待値は妥当な責任準備金とは見なされていない。言い換えれば、そのような責任準備金はソルベンシー不足だというのが一般的な認識である。一般的な意味での責任準備金を積ませるには何らかの意味での安全割増が考慮される必要がある。破産時のペナルティに応じてエージェントは破産を平均以上に避けるようになるため、何らかの意味での安全行動～この場合は安全割増を考慮した責任準備金を積むようになると期待される。
- ・ところが、この破産時のペナルティとしてどのくらいが妥当かが分からぬという問題がある。破産時ペナルティと実際に形成される安全割増の関係は多くの場合単純ではないと考えられる<sup>28</sup>。そもそもこの関係は少なくとも方策オンの場合はアルゴリズムにも依存する。このため、一

<sup>27</sup> 位相構造について何も言っていないので、局所解というのは数学的には少し不正確かも知れない。厳密には破産する前提での最適解ということになる。

<sup>28</sup> 特に、責任準備金の情報を直接は一切用いないという制限があるので尚更難しい。

一般的に妥当だと考えられている水準の安全割増に相当するペナルティを導出するのが困難という問題がある。

・1つの方法として、その時点の実際の支払保険金×120%の責任準備金が積まれていなければ破産扱いにしてしまうといったものが考えられるが、これは1年分の支払しか考慮していないで長期支払能力を確保するという観点からは問題があるものと考えられる。どちらかといえばVaRに近い性質のものだろう。

・あるいは別の方法として、責任準備金計算用には負債キャッシュフロー×120%にしたものに入力するといったものも考えられるかもしれないが、学習上は結局どういう条件のときに破産するかだけが考慮されるため、実はこの方法は同じ結果に陥る。

・いずれにしても現行の責任準備金にしろ、安全割増にしろ、VaRにしろ、「破産しないために必要な額」についてであって「破産したらどうなるか」については何ら語っていない。標準的な強化学習の文脈、即ち期待値ベースでは、これらをペナルティのベースに直接据えることはできない<sup>29</sup>。そもそも破産を考慮に入れるとなるとゴーイングコンサーンから外れてくるので、標準的な会計基準がどこまで妥当かという問題もあるかもしれない<sup>30</sup>。この問題は解決が難しいと考えられるため、本論文ではこれ以上扱わないことにする。

## ■標準アルゴリズムによる結果

・図19にこれまでに使用してきた標準的なアルゴリズムであるDQN・A2C・DDPG・NAF・AWR・ES・GAによる結果を示す。積立に成功した場合と失敗した場合で利得が大幅に異なる（破産ペナルティが発生するかしないか）ため、学習曲線の変動が非常に激しく見づらいことから、ここでは箱ひげ図を用いている。

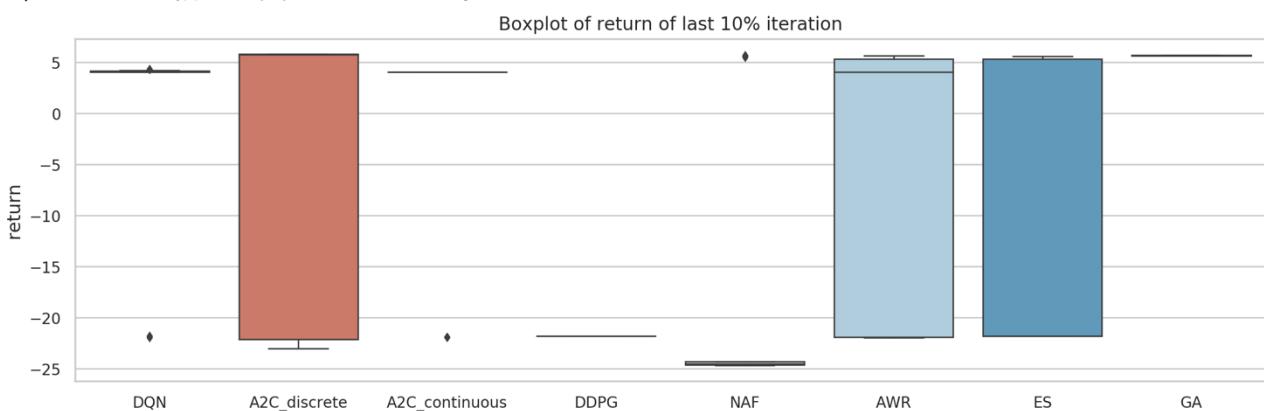


図19: 学習曲線の最終10%部分における箱ひげ図：ほとんどのアルゴリズムでは破産している（-25）か、破産したりしなかったりであり、安定して破産を避けられているのはGA（右端）のみであることが分かる。

・GAを除くと安定しては積立できず、DQN・A2C・ES・AWRは積立に成功したり失敗したり、DDPG・NAFは常に積立に失敗していることが分かる。多くの場合、開始直後に責任準備金をゼロ付近に設定して（資産を全額配当に出してしまう）、10年目の保険金支払いを踏み倒してペナルティを受けている～つまりは局所解に収束していた。また、ここでは示していないが、ハイパーパラメータ/ネットワーク構造などを色々と試しても結果は同様であった。

## ◇何故破産するか？

<sup>29</sup> 安全割増は理論的には破産に対する（社会的なものも含んだ）何らかの意味でのコストを反映していると思われるが、強化学習の行動評価基準である期待値アプローチに単純翻訳することは難しいと思われる。例えば、分布が正規分布だったとき、 $VaR[X] = E[X] + 2.77\sigma = E[X + \text{破産ペナルティ}] = E[X] + E[\text{破産ペナルティ}] = E[X] + 1\% \times \text{破産ペナルティ}$ となる。つまり、 $\text{破産ペナルティ} = 277\sigma$ になるのだが、これは $\sigma=0$ だとゼロになってしまう。つまり、確定債務なら踏み倒して良いということになってしまう。

<sup>30</sup> そもそも払えなくて破産しているので、即時支払しなければならない課徴金を課すという対応は現実的にありえないようと思われる。

・破産しやすい理由は理論的に分かる。離散系と連続系で事情はやや異なるので各々説明する。先ず離散系の場合、行動として責任準備金を0から20までの21通りの中から1つランダムに選択する。責任準備金は常に（概ね）10なので、1度でも10未満を選択すると最終時点で破産する。エピソード長は10なので、破産しない確率はランダムに行動した場合概ね $1/2^{10}$ であり、これは概ね1/1000である。このため探索経験のほとんどは破産しており、経験メモリあるいはエージェントの行動は破産経験で埋め尽くされ、NNの更新も基本的には破産前提での最適化を行うようになる。すると、既に述べたように、初期時点で全額を配当流出させるという局所解へと向かっていってしまうものと考えられる。

・次に、連続系の場合は、仮に妥当な責任準備金を学習したとしても、標準的な強化学習エージェントは上下対称に探索を行うため、各時点で $1/2$ の確率で責任準備金の積立不足に陥ってしまう。1度でも積立不足になるとリカバリーできないので、先ほどと同様に連続系でも最適行動に到達後に責任準備金を積める確率は $1/2^{10}$ の程度であるので結局は同じ状況になってしまう。但し、これは最適解に一旦至った場合の話であって、方策オンの場合は最適解に至る前に留まることもあり、その場合は破産はしないのだが、目的達成できていないことに変わりはない。ハイパーパラメータ次第で実用上問題無いレベルに停留するかもしれないが、そのような方法に実用性があるとは思われないため見送った。

・更に、いずれのケースでも破産判定は最終時点でのみ行われるため、行動の成否が環境からエージェントに伝達され、それがNNの更新を通じて前半の行動へと反映されるには非常に長い学習時間が必要である。これはTD(0)などのステップ数の少ないアルゴリズムでは非常に厳しい。

・破産前提での解が大域解に近ければ実用上問題ないという考え方もあるかもしれないが、この解は開始直後に全額配当して破産するというものであった。つまり、適切な責任準備金の保持という大域解からは程遠く実用性はまるでないである。

・以上の通り、そもそも破産しやすく、破産情報が遅延して発生し、学習できてもほぼ確実に破産し、破産すると最適解から遠い局所解に収束する、という状況になっている。

#### ◇技術的安全割増

・ところで、ストレスを狙いに行くことで破産するわけだから、何らかの安全割増を反映した責任準備金を積ませることでストレスに行かないように誘導するということが考えられる<sup>31</sup>。しかし、現在考えている環境は確定債務であり確率的要素は一切無いので、この安全割増はあくまでこの技術の限界を示すものになる。

・環境にノイズを加えて確率的にして破産をソフト化（オプション価格がボラティリティによって滑らかになるのと同じ）することが考えられるが、そもそも確定債務であるものを純粋に技術的な理由で確率的にする場合、どの程度が許容されるのかよくわからない。最初はノイズの多い環境で学習させて段々探索レベルとノイズレベルを下げていくという方法も考えられるが、非常に計算時間がかかるものと思われる。しかし、方向性としてはありかもしれない。

#### ◇GAとAWR

・GAは破産しづらい。これは、GAではエピソード単位でのみ性能が評価され、更には過去から最も性能の良かったエージェントが保存されるため、破産による報酬遅延に強い上に基本的に性能の劣化を引き起こすことがなく、1度でも運良く破産しないエージェントが生成されれば少なくとも破産はしなくなるためである。とはいえ、そのエージェントから生成されるエージェントはやはり大半が破産することから、サンプリング効率そのものは悪く、妥当な責任準備金を計算することは難しい。

・AWRもGA程ではないが破産をある程度回避できている。これはAWRがGAと同じくエピソード単位で方策評価を行う為、先に述べた報酬遅延問題に対して強いためと考えられる。但し、最善エージェントの保存は行われないため、GA程ではないのだと考えられる。

・以上を考えると、報酬遅延問題に強いエピソード単位での評価・破産しないエージェントあるいは経験の保持が性能改善の方向性として見えてくる。

<sup>31</sup> 先程の安全割増の議論は責任準備金として妥当な安全割増を積ませたいという話であって、ここの安全割増（？）はあくまで技術的な安定性の話なので異なる。

## ◇報酬体系変更

- ・解決策だが、先ず強化学習の王道として、報酬体系の変更が考えられる。
- ・報酬体系の変更時に気をつけなければならないのは、責任準備金を外挿して使うような形にしてはならないということである。責任準備金を（自動）学習させたいのだから、責任準備金を直接報酬に埋め込んでしまっては意味が無いわけである。また、この先の展開として、保険料・資産運用などと結合させた会社モデルを作りたいので、あまりにも責任準備金に特化した、他との結合が困難になるような報酬体系も避けなければならない。以下、幾つか紹介する。

- A) 破産時に足りない分のキャッシュをペナルティに追加する：10年目と11年目にそれぞれ100の保険金支払がある場合を考える。このとき、仮に10年目に99資産が残っていた場合のペナルティは $100-99=1$ である。これに対して101持っていた場合のペナルティは10年目はそのまま支払い11年目では $100-1=99$ となって後者の方がペナルティが大きいというおかしな結果になってしまう。これは足りない分の単調増加関数をペナルティに加算する限り同様の問題が生じてしまう。このため、この方法は一般的なケースには拡張できない。
- B) 破産時にその後の全ての保険金支払の合計額 - 残存資産額をペナルティとして付加：初回保険金支払のタイミングが遅い場合、どの時点で責任準備金をゼロに設定したとしてもペナルティは同じである。この場合、より早い段階で責任準備金をゼロに設定する方がハードルレートの関係上有利になる。このため局所解の方へ収束する。一方、責任準備金を保持しようとする方向の力も働くと考えられるが、両者の綱引きになり不安定と考えられる。
- C) 借入を可能にして破産を防ぐ：開始直後に全額を配当として流出させ、保険金支払いのタイミングで借入を行う方が、借入金利の適用される期間よりも開始時点から保険金支払いが発生するまでの期間の方がずっと長いので、借入金利がハードルレートより高くても初期破産が最適戦略になる。これを防ぐためには保険金支払の発生時点に合わせて順序が逆転しないような借入金利を別途設定する必要がある。このような金利を個別に計算するのはキャッシュフローが複雑化すると困難だと思われる。
- D) 最初にハードルレートをマイナスに設定して責任準備金を多めに積ませるように学習させ、その後にハードルレートを本来の値に戻して調整する：結局は最終的な最適解が不安定であるため同じ結末に至ってしまう。離散系の場合はうまくいくかもしれないが、最終的には連続系に適用できなければ意味がない。
- E) 責任準備金保持にボーナスを持たせ、破産しなかった場合は最終時点で全額回収する：この方法は**責任準備金ボーナス法**と呼ぶことにして別途後述する。
- F) そもそも責任準備金は考えない。エピソード中の即時報酬は全て無視し、最終時点の全保険金支払後の資産のみを即時報酬として考える：途中で新規契約を獲得していくようなケースを扱うことができなくなる。また、収益機会の無いときに配当としてリリースできないのでROAの低い解になるかもしれない。破産という不連続点は発生しなくなるが、即時報酬が最終時点でしか発生しないので報酬遅延問題が発生する。この方法は検討の余地はあるが、本節の枠内で論じても仕方が無いので、以後触れないことにする。

- ・問題として大きいものとして、破産時にペナルティを課す場合、そのような責任準備金に至った経緯をMDPの枠組みでは考慮できないことが挙げられる。このため、破産前提の場合、開始時点で全額流出させるのが最適解になりやすい。
- ・以上のように様々な報酬を考えたが、Eを除いてどれもうまく行きそうになかったり、うまく行かなかった。そこでアイデアを得るべく以下に述べる逆強化学習による報酬関数の逆推定を行うことにした。今回の場合、最適行動（正しい責任準備金を積む）は既に分かっているため、これを元にして逆強化学習で報酬体系を逆算させれば、人間が考えつかないような報酬体系が出てくるかもしれない、ということを期待したのである。

## ■逆強化学習

・逆強化学習 (inverse reinforcement learning : IRL) とは強化学習の一分野で、通常の強化学習では環境と報酬が与えられた上で最適行動を学習するのに対し、逆強化学習では環境と行動が与えられた上で、その行動が最適行動になるような報酬を学習（逆算）する<sup>32</sup>。通常、そのような報酬は複数考えられるため<sup>33</sup>、何らかの制限をかけることになる。

・本論文では最大エントロピー法 (4)・ベイズ法 (5) の 2 種類を試した。各々の方法は以下の通りの事後分布を仮定している。尚、原論文では即時報酬は推移先の状態のみに依存し現在の状態や行動には依存しないとして定式化しているが、配当の大きさは保有資産（つまりは現在の状態）に依存するため、本論文ではこれを少し拡張して状態と行動に依存して即時報酬が決まるようにしている<sup>34</sup>。

$$P(O_\chi | R) \propto \exp \alpha \sum_i R(s_i, a_i, R) \quad : \text{最大エントロピー法}$$

$$P(O_\chi | R) \propto \exp \alpha \sum_i Q_*(s_i, a_i, R) \quad : \text{ベイズ法}$$

・ $P(O_\chi | R)$ ：報酬体系  $R$  において最適行動としてトラジェクトリ  $\chi$  が生じる確率

・ $Q_*(s_i, a_i, R)$ ：報酬体系  $R$  における最適行動価値関数

・ $R(s_i, a_i, R)$ ：報酬体系  $R$  における即時報酬

・ $s_i, a_i$ ：エキスパートのトラジェクトリ中の状態と行動

・結果は図 20 の通りである。尚、ここでは計算を早く収束させるために、10 年ではなく 5 年として、保険金額も 5 とした。いずれも現在の資産が 4 以下のときは 0 のところに、5 以上のときは 5 のところに縦線状にプラスの即時報酬が発生している。つまり、双方共に責任準備金 5 を正しく積めば即時報酬がそのまま発生し、足りなければ即時全額流出させるとやはり即時報酬が発生する、という身も蓋もない結果になっている。というわけで、残念ながらこの報酬体系からは当初の目的である問題の解決に繋がるような報酬体系の構築は難しいと考えられる。

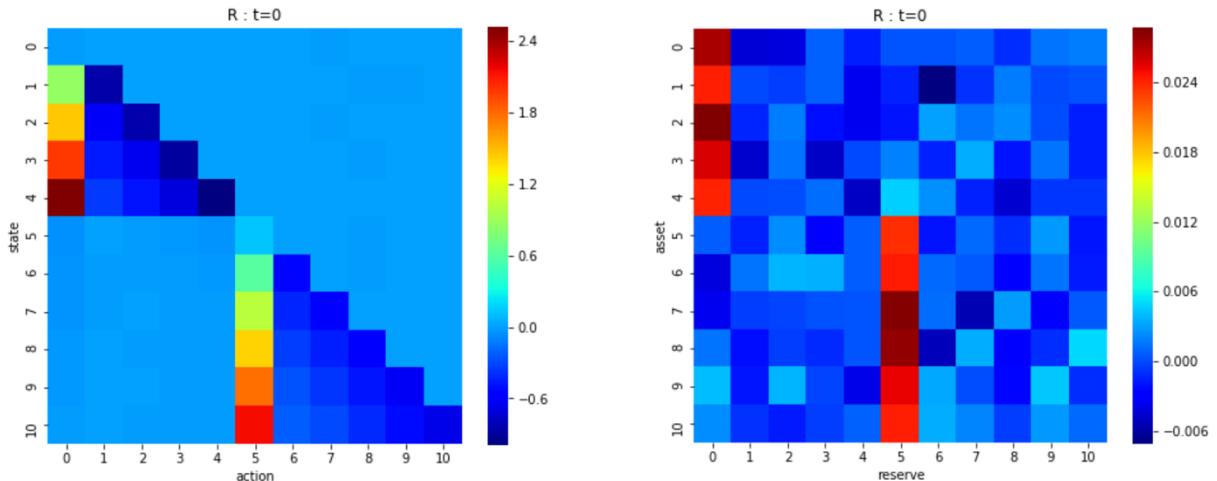


図 20: 逆強化学習を用いた即時報酬の逆算。最大エントロピー法（左）とベイズ法（右）：いずれも X 軸が現在取る行動（責任準備金設定額）、Y 軸が現在の資産額である。開始時点 ( $t=0$ ) での結果のみ示しているが、他の時点でも傾向は同様である。資産額が 4 以下の場合は行動 0 を取るところへ、資産額が 5 以上の場合は行動 5 を取るところへ即時報酬が集中していることが分かる。正しい責任準備金は 5 であるので、これは正しい責任準備金を積めば即時報酬が発生するという結果になっている。

<sup>32</sup> 例えば、優秀な営業販売員の行動からコミッション体系を逆強化学習で生成して、他の営業販売員が自然と真似をするようにするようなものである。

<sup>33</sup> 例えばスカラ一倍。

<sup>34</sup> つまりは通常の強化学習の状況設定に概ね等しくなっている。ただ、この部分に関して理論的検証はしておらず、あくまでアイデアを得る目的で行っている。

・しかし、これは逆強化学習によって、保険契約に基づく金銭授受という最低限の事実とその際の最適行動から、利益認識体系が構築されたと見なせるかもしれない。エントロピー最大化を「最も分かりやすいシンプルな体系を選出する」とナイーブに捉えるならば、これは逆強化学習によって「正しい行動を促す分かりやすい行動指標」が生成されたと見なせる。今回は強化学習で責任準備金を積めないのが問題となっているが、仮に強化学習で積めるのならば、最初の困難な問題に対して強化学習で最適行動を学習し、その最適行動に基づき逆強化学習でエントロピーが最大になるような報酬体系を構築することで、人々が解釈しやすい、従うべき報酬体系としての会計基準的なものが生成されるのかもしれない<sup>35</sup>。

## ■層化経験再生

- ・問題のうち、「破産経験が多すぎるために、破産前提での最適化が行われてしまう」点を解決するために、経験再生を層化して、学習時の経験を、破産したものとしなかったものと均等に扱うことが考えられる。本論文中ではとりあえずこれを SER (Stratified Experience Replay : 層化経験再生) と呼ぶことにし、経験再生を持つタイプのアルゴリズム (DQN・DDPG・AWR) に適用することにする。
  - ・通常の経験再生では、経験は発生の都度そのまま経験メモリに蓄積されるが、図 21 に示すように、SER ではエピソード終了まで中間メモリに蓄積される。そして、エピソード終了時に破産していれば破産メモリに、そうでなければ非破産メモリに経験を格納する。そして、学習時には破産メモリと非破産メモリから規定の割合（例えば 50%50%）で経験を抽出して NN の学習に用いる。尚、付随してアドバンテージ (n-step) を導入した。

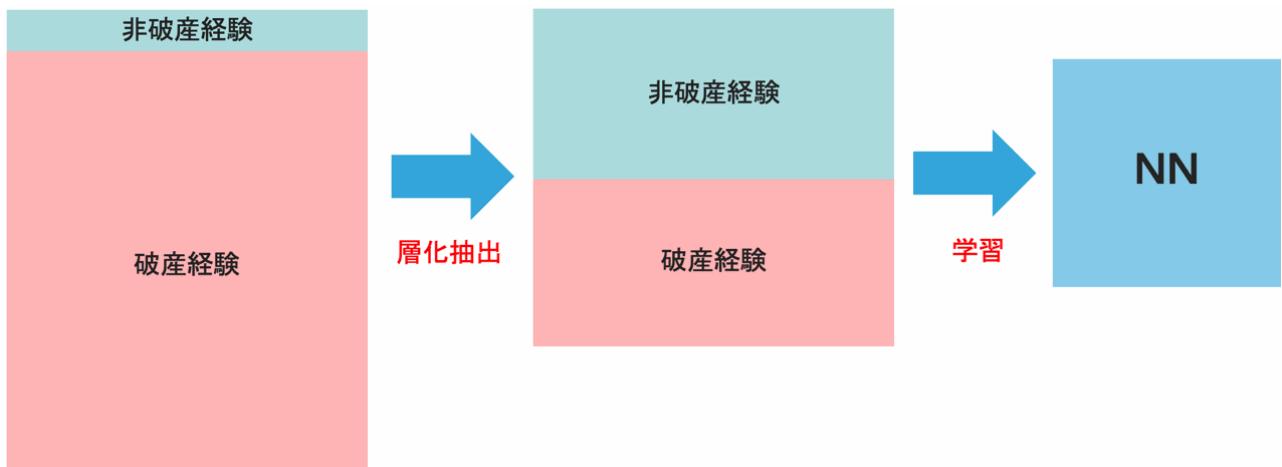


図 21: SER のイメージ図：経験メモリから直接 NN に渡さずに、層化抽出を行ってから渡す。

- ・結果は図 22 に示すとおりである。DQN では劇的に改善しているが、DDPG・AWR ではイマイチである。特に DQN の場合、適用前後の Q 関数の進展をヒートマップ化すると、図 23 に示すように大幅な変化が見られる。適用前（上）は 8 割りがた学習してようやく責任準備金 10 の境界を認識し始めているが、SER 適用時（下）には 2 割程度の非常に早い段階からこれが認識されていることが分かる。

<sup>35</sup> 勿論、実際に会計基準と呼べるためには、少なくとも、任意の時点・事象において現金移動とトータルあるいは現在価値で一致していなければならないと思われる所以、単純な一問題の最適化のみから会計基準とまで言うのは言いすぎではある。しかし夢のある話だとは思われる。

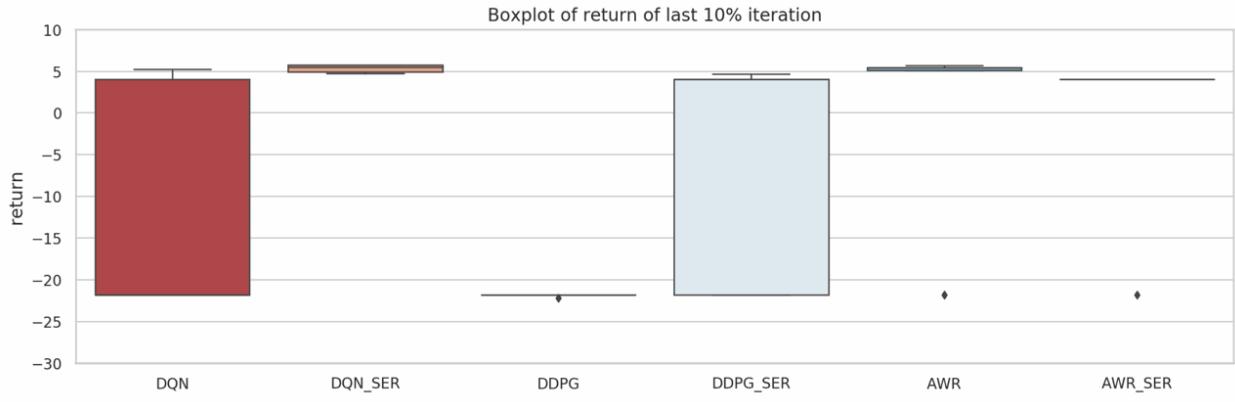


図 22: SER を適用した場合の各アルゴリズムの性能：基本的には改善傾向にあるが、DQN 以外はなんとも言えない。

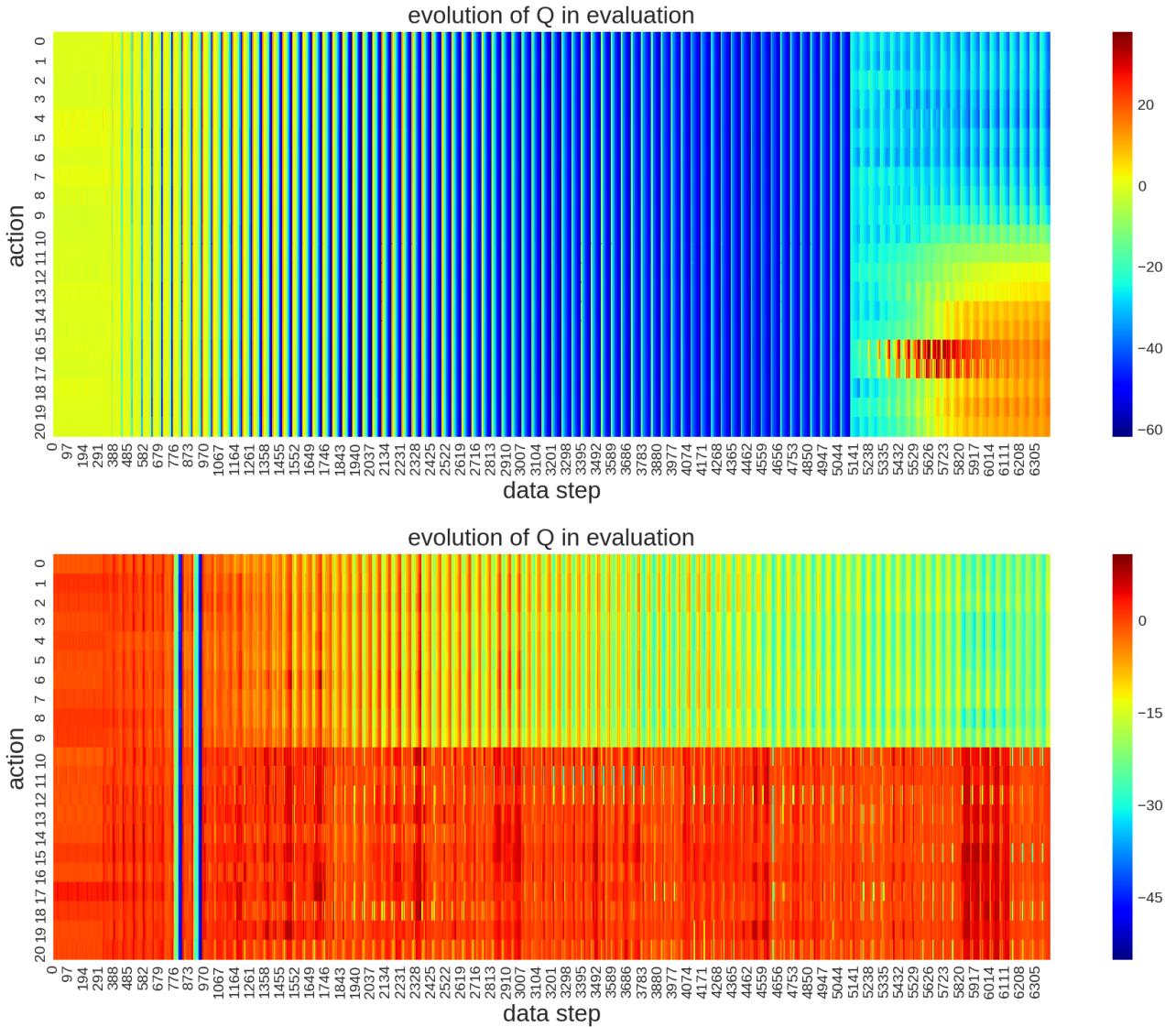


図 23: SER 適用前後の DQN の Q 関数：上段が適用前、下段が適用後である。X 軸が学習ステップ数、Y 軸が各時点における行動である。実際には様々な状態において Q 関数を計算しているため、同条件で比較しているわけではないが、本問題では責任準備金は 10 未満だと必ず破産するため、どの時点でも 10 を境にして Q が大幅に変わることになる。適用前では長い範囲に亘り 10 を境にする Q の変化は見られず、最後の方（右端）でようやく変化が見られる程度である。これに対して適用後では早い段階から 10 を境にして変化していることが分かる。尚、この図の前提は前図の計算に用いたものとは異なっているが傾向は同じである（異なった前提での結果を用いている理由は上段の SER 無しにおいて非破産解に至るかどうかが運次第であり、前図の試行では非破産に至らなかったためである。つまり、上段は上手く行った場合でもこれぐらい学習が遅い、ということである）。

・ **DQN** の場合、SER を用いることで破産に至ることはほぼ避けられるようになるものの、かといって最適解にはなかなか収束しない。この問題における責任準備金は 10 だが、DQN エージェントは実際にはこれより大きな値を積んでしまっていることがほとんどであった。これは選択肢間の順序・位相関係が考慮できないためと考えられる。実際、経験メモリに蓄積された経験を確認すると、非破産経験の数はランダム行動の際とあまり変わりない。これを解決するためには、離散でも選択肢間の順序関係を考慮するタイプの NN を用いることが考えられる。尚、探索方策としてボルツマン方策を用いても目立った改善は見られなかった。

・ **DDPG** の場合、SER を用いてもあまり効果は無い。これは方策の更新が勾配ベースであるため、仮に価値関数を学習できたとしても一旦破産側に落ちてしまうと（その確率は 99.9% だった）勾配によってそのまま局所最適解に必然的に至るからであると考えられる。そのため、方策の更新には勾配ベースの方法を用いず、ランダムサンプリングを用いる方法も試したが、そのようにしても破産するかしないかはかなり運次第であり、DQN のように安定して破産しないようにすることはできなかった。

・ **AWR** の場合、SER を用いてもあまり効果は無い。これは元々 AWR がエピソード単位で学習するアルゴリズムであるため、報酬遅延問題に対して強いことが考えられる。図 22 のデータ上ではわずかに効果が出ているように見えるが、ハイパーパラメータを様々に変えたところ同程度か若干悪化しているようにも見受けられた。

## ■責任準備金ボーナス法

・前述した報酬体系である責任準備金ボーナス法を適用した結果について述べる。責任準備金を積んでおくとハードルレートを僅かに上回るボーナスが即時報酬に加算され、破産したらそのまま、破産しなかったらその終価がマイナスの即時報酬として発生することでキャンセルされるという報酬体系である。これにより、破産する場合は責任準備金をなるべく多く積むのが最適解になり、破産しない場合は責任準備金をなるべく少なく積むのが最適解となる。数式は以下の通りとなる。

$$\text{CumRB}_t = 0$$

$$RB_t = \text{reserve}_t \times (\text{Hurdle} + 0.01)$$

$$\text{CumRB}_{t+1} = \text{CumRB}_t \times (1 + \text{Hurdle}) + RB_t$$

$$RB_T = \text{CumRB}_T \text{ if not bankrupt else } RB_T$$

- ・ CumRB : 累計責任準備金ボーナス

- ・ RB : 責任準備金ボーナス。当期の即時報酬に加算される

- ・ reserve : 責任準備金

- ・ T : 満期時刻

・これは、イメージとしては概ね図 24 のようになっている。先程と異なり、局所解が責任準備金のゼロ点ではなく、大域解と同じ位置に移動していることが分かる。但し、この図は正確ではない。この新しい報酬における破産前提の最適解は、過剰な責任準備金を積み続けた挙げ句に保険金支払い直前に僅かに足りない責任準備金を残して破産することであるため、大域解とは一致しない。あくまで破産前提でも責任準備金を多く積んだ方が利得が多いというイメージに過ぎない<sup>36</sup>。

---

<sup>36</sup> 1つ特に注意した方が良い点として、局所解と大域解との距離が挙げられる。大域解は責任準備金ボーナスの有無に拘らずに常に 10 を積み続けることだが、局所解はボーナス法でない場合は最初に 0 を積んだ後は任意、ボーナス法の場合は最後まで 15~20 を積んだ後に最後だけ 9 にするというものである。ここで重要なのが、資産額以上の責任準備金をセットしても資産額でキャップされるという仕様である。ボーナス

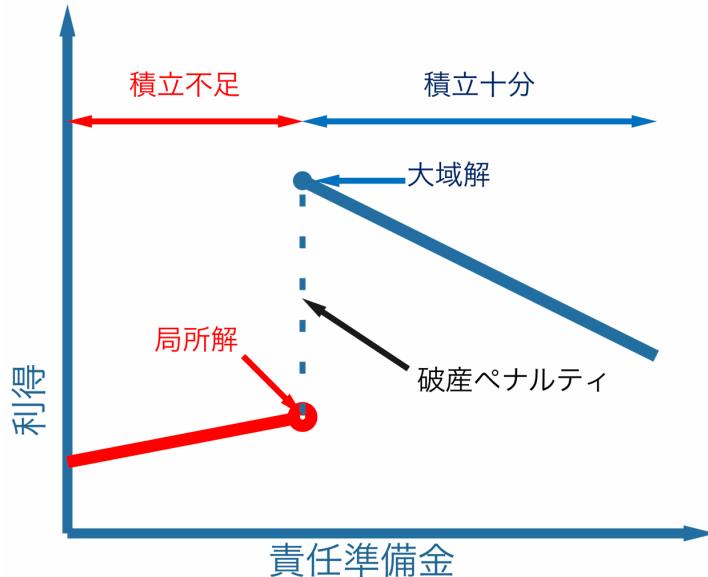


図 24: 責任準備金ボーナス法における責任準備金と利得との関係のイメージ：先程と異なり、積立不足で破産前提だとしても責任準備金を多めに積む方が利得が高くなることをイメージしている。

#### ◇ノーマルアルゴリズム

- ・図 25 に結果を示す。ほぼどのアルゴリズムでも破産していないことが分かる。しかし、最終性能にはかなり差があり、多くの場合責任準備金を積み過ぎになることが分かった。
- ・恐らく順序としては、最初に破産経験が大量に蓄積されるため、その最適解として責任準備金が増えていく。破産しないためには全ての時点で責任準備金が 10 以上に設定されている必要があるので、非破産に至る時点で平均的に積まれる責任準備金は 10 よりも幾らか大きくなっている。しかし、このまま非破産フェーズに入っても、ストレスの非破産経験はサンプリング効率が非常に悪いことは前述の通り変わらないため、結局はそこから学習が進行しづらく、破産期に学習した多めの責任準備金が更新されずに残ってしまうと考えられる。
- ・破産しないこと自体は「破産するよりはマシ」に見えるかも知れないが、積み過ぎでいいなら最初から計算させなくてもいいという立場もありうるので、この方法に絶対優位性があるかといえば、そうでもないとも考えられる。

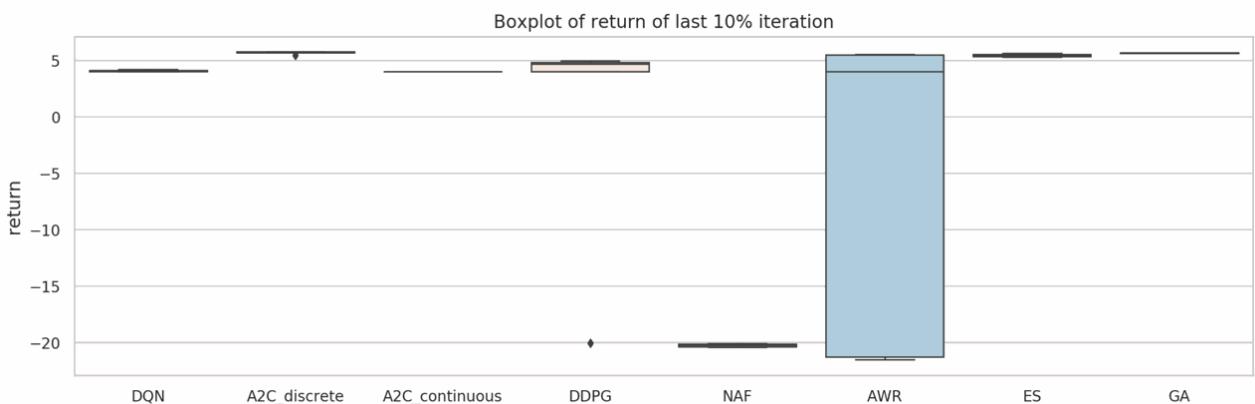


図 25: 責任準備金ボーナス法に基づく結果：NAF と AWR 以外は全て非破産になっていることが分かる。

無しの場合、最初に責任準備金を 0 にしてしまえば後は幾らにしようとも 0 でキャップされるため結果には影響しない。つまり、最初に 0、その後に 10 を積み続けるのも局所解になっている。この解と大域解との差は最初が 0 か 10 かでしかなく、距離にして  $(10)^{1/2} = 3.16$  である。これに対してボーナス有りの場合、局所解は常に 15 以上を積み続けるため、大域解との距離で考えると最低でも  $(5^2 + 9 + 1)^{1/2} = 15.03$  であり、明らかに大きくなっているし、変わらなければならない時点も全時点に及んでいる。つまり、最終的な解の距離のみを比べればボーナス法の方が遠くなっているのである。

### ◇層化経験再生と組み合わせた場合

- ・図 26 に SER と組み合わせた場合の結果を示す。残念ながら性能は特に向上していないか、若干悪化するケースが多かった。特に DQN-SER の場合、学習が不安定になり明確に悪化することが分かった。SER と責任準備金ボーナス法は相性が悪いということだが、理由は分からぬ。

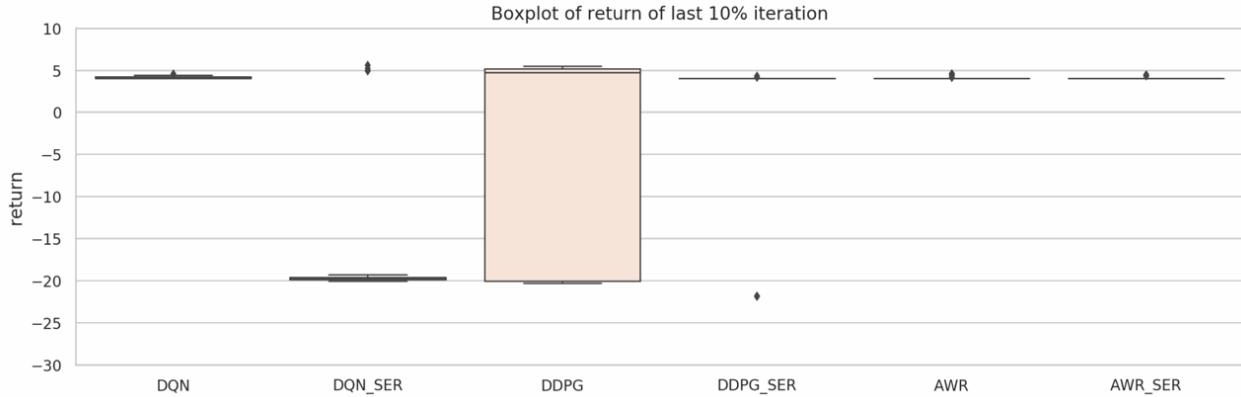


図 26: SER と責任準備金ボーナス法に基づく結果：DQN\_SER が破産していることが分かる。

## ■外挿化

- ・以上のように、責任準備金をキャッシュフローのみから自動生成するのは現行の標準的な強化学習ではかなりハードルが高いと思われる。責任準備金ボーナス法・GA・AWR・DQN-SER では破産そのものはある程度避けられているが、妥当な責任準備金の生成までには至っていない。月単位で長期の問題を扱う場合は時間ステップ数が 1200 (=12 ヶ月×100 年) 程度まで伸びるため、ランダム探索では不可能だと思われる。これをうまく扱うには探索手法そのものの改善が必要だと思われるが、一般性を維持したまま実行可能な方法があるかどうかは分からぬ<sup>37</sup>。

・また、仮に強化学習で責任準備金を生成可能だとしてもその詳細や内部構成が外部からは分からぬという問題がある。例えば、責任準備金のうちのどれだけがベストエスティメイト相当で、どれだけが安全割増に相当している部分なのかはわからない（最も、これに関しては学習後にシミュレータ上で検証することは可能である。あるいはデュエリングネットワークのような構造もあるかもしれない）。更に、拡張して、コミッショニングボーナス・新規事業投資などまで含めた場合、配当として流出させずに残してある部分は責任準備金相当なのか、あるいは将来の投資のために残してある部分なのか見分けがつかない。結局、本節の責任準備金は営業保険料式に近いものだが、実際には将来の事業投資などまでも含めた広い概念になっているため、どの程度までが所謂負債になっているのか分からなくなる。これは、シミュレータ上の行動を確認することで事後的にその理由を解釈できるかもしれない（この事業を開始するためにこれだけの資金を積み立てていたのだな、など）。しかし、人間に認識しやすい大きなイベント・パターンが発生していないとその理由はわからないかも知れない。

- ・というわけで外挿した方が良さそうということになるわけだが、少し問題がある。
  - ・先ず、強化学習で全てを統一的に扱うという方針から外れることである。責任準備金というある種の中間的なチェックポイントをマニュアルで設定するので、そのためのコストがかかる。ただ、これはある程度は気持ちの問題だし、全てを強化学習でやるのが現実的でないというのが比較的一般的な認識のようなので、さしあたり目立った問題ではないと思われる。
  - ・次に、一般的なバリュエーション手法は会社行動を動的には反映していないことがある。例えば、バリュエーション上の解約率が 3% ぐらいになっていた場合、各時点での外挿された責任準備金計算はそれに基づき行われるが、実際の強化学習エージェントの行動は将来的な解約増をもたらすものになっているかもしれない。この場合、外挿されているバリュエーション前提と実際の強化学習エージェントによる行動結果が乖離する。これは JGAAP 責任準備金のようにロックイ

<sup>37</sup> 様々な探索手法については例えば [6] を参照のこと。

ンかつ比較的保守的に設定されていれば問題無いが、IFRSのようなロックフリータイプや料率変動型など責任準備金前提が変更される場合にはミスマッチを起こしうる。毎年基礎率不一致による差異が発生するし、単純にソルベンシー不足に陥るかもしれない。あるいは、外挿されたバリュエーションアサンプションで早期に利益を認識して、バリュエーションアサンプションとの乖離による調整ロスを後期に認識させた方が割引率の関係上EVが上昇するため、エージェントがこのような方策を採ることは十分に考えられる。

・現実にはロックフリーの場合、バリュエーションアサンプションは実績に追随して更改されるが、これは将来に対する洞察を含むので、シミュレータにそこまで入れるというのは現実的でないと思われる。この点を解消するには、エピソード単位で責任準備金を事後的に計算して、バリュエーションアサンプションを少しずつアップデートしていくべきかもしれない（責任準備金の計算メカニズム自体がNNで表現されれば特に難しくないと思われる。実際、AWRはバリューネットワークに関してそのようなアルゴリズムである）。これはある種の動的なモデル構築によるモデルベース強化学習かもしれない。

・他には、**模倣学習**などを用いて通常の責任準備金を一度学習させた上で強化学習オンリーでチューニングすることも考えられる。あるいは、**カリキュラムラーニング**を用いて期間の短いものから長いものへと徐々に問題の難易度を上げていく方法も使えるかもしれない。いずれにしても妥当な責任準備金が不安定解ならば結局は失敗するため、不安定解にならないような報酬体系・探索アルゴリズム・NNの更新方法は必要である。

## ■まとめ

- ・標準的な方法では責任準備金を積むのが難しいことを確認した。
- ・逆強化学習をそのまま使っても実質的に責任準備金がそのまま示されるだけであった。
- ・層化経験再生・責任準備金ボーナスを使うことである程度は破産を軽減できることが分かったが、確実ではない。
- ・責任準備金を強化学習単独で積ませるのは止めた方がよさそう。

## ■7.考察

### ◇全般

・一般的にアクチュアリーはプロジェクトモデルを持っているので、強化学習でスタンダードになっているモデルフリーの方法に拘泥する必要は無いと思われる。モデルベースの方法の方が効率が良いと考えられるため、今回は検討しなかったがモデルベースの方法をあたる方が筋が良いかもしない<sup>38</sup>。一方、第3節で紹介したように一切モデルを必要とせず、実データを経験再生に投入して完全にゼロベースで価値関数などを構築する方法もあるので、究極的にはこの方法になるかもしない。とはいうものの保険会社においてこの方法が機能するレベルでのデータ収集・行動調整が可能になるのは大分先だろう。いずれにしてもこのようなモデルベースの方法があるとするならば、それはアクチュアリーの力なくして不可能と思われる。

### ◇精度面など

・安定性・再現性の無さなどがしばしば指摘されている(6)(7)。実際、深層強化学習の研究の多くは安定性の確保を目的としている。また、近年では標準的な評価環境についてDeepMind等から提案が出ていたりしている。本研究においてもハイパーパラメータにより性能が極端に変わることが多々あった。学習速度が多少変わる程度ならば実用上は甘受できるかもしれないが、学習の崩壊（責任準備金で見た破産のような事象など）を起こすようだとかなり厳しい。仮に業務で用いるならば、保険業務特有の学習の崩壊の定義・検出・傾向・抑制・ボトムラインへの影響の、体系的・分野及び会社間横断的な研究が必要と考えられる。

・本論文で見る限りでは、特に連続系で精度があまり良くない印象がある。金融業務の計算の多くは計算手順が定式化・標準化されており、比較的精度の良い解を求めることが可能である（あるいはそういうものしか商品・バリュエーション手法として定着していない）。この解に基づいたプライシング・バリュエーションが既に業界のスタンダードであるため、強化学習の生成するプライシングでは負ける可能性が高いと考えられる。既存手法で精度が出せる計算に関しては極力強化学習に担当させないようにする必要がある。ただ、本論文ではそうした場合に強化学習の適用領域がどの程度残っているのか、あるいは新規に開拓されるのかまでは検討できていない。

### ◇前提

・需要関数が分からない。生保会社のランザクション・価格改定の頻度ではこれらを広範囲に渡って精確に推定するのは難しいと考えられる。営業力・ブランド力などのソフト事項も難しい。  
・アサンプション（計算前提）に関しては外挿の比率が大きい。特に歴史のある大手社以外は自社経験のみで全範囲における死亡率・発生率・解約率を作ることは明らかに不可能であるし、どんなに強力なアルゴリズムを使用しても統計的に安定したアサンプションを生成するまで会社の資本や評判が持たないだろう。  
・長期に関しては責任準備金計算が難しい上に、そもそもアサンプション信頼性の観点から実用上の意味があるのかどうか自体が疑わしいと考えられる。そのため、シミュレーション自体は数年～10年程度の短期に限り、終了時点の純資産額で評価するなどした方が良いと思われる。但し、会社価値をどう評価するかは言えないところもある。純資産のみならず、保有の大きさなどで評価する向きもあるだろう。そのような場合は複数タスクを同時に学習させるタイプの方法が良いかもしれない。

### ◇雑感

・AIが運用するファンドは極めて複雑なデリバティブと見なせるが、どうプライシングされる（されている）のだろうか？経済理論的な整理はどうなっているのだろうか？  
・仮に量子コンピュータなどで即座に解けるようになったとしても解決しない部分が多いと思われる。大部分は利用できるデータが少ないとによる。これは仮に多少なりとも顧客タッチポイントを増やしたとしても原理的に解決しないかもしれない。生命保険業という本来ビッグデータ

<sup>38</sup> 尚、本論文には掲載していないが、AlphaZeroを責任準備金問題に適用してもよい結果は得られなかった。しかし、時間の関係上、デバッグ・分析が十分ではないので載せなかった。

ビジネスであるものが、IT・IoT社会になるに及んで相対的にトランザクションが少ない故にビッグデータ・AI時代においては逆に相対的にスマートデータビジネスになるという現象が生じているかもしれない。あるいは生命保険ビジネスは長期契約という前提 자체が崩れるかもしれないし、生命保険はビッグデータ潮流に乗らなくてもいい安定なビジネスということかもしれない。このあたりは損害保険や少額短期の方がフィットする領域なのかもしれない。

・どのくらいトランザクションデータを増やしたら、どのくらいタッチポイントを増やしたら自動化が可能なのか、それはどの程度の精度なのか、コストに釣り合うのか、といった試算が可能かもしれない。

## ■A.付録：保険商品の収益性のアサンプション詳細感応度の自動導出

### ■概要

- ・2019年現在、TensorFlow<sup>39</sup>・PyTorch<sup>40</sup>・Chainer<sup>41</sup>といった幾つかのディープラーニング用フレームワークが存在するが、これらは基本的に自動微分機能を備えている。というのも、ディープラーニングのオプティマイザは勾配降下法をベースにしているからである。
- ・これらのフレームワークはディープラーニングを想定して設計されてはいるが、自動微分機能そのものはディープラーニングに限らず利用可能であるし、行列計算や各種の関数が利用可能で生保数理のかなりの部分は直接カバーできる。つまり、生保数理の対象はこれらのディープラーニングフレームワークで自然な形で扱うことができる<sup>42</sup>。イメージとしては、例えば潜在価値(EV)の場合、図27諸々一式をディープラーニングフレームワークで一貫して構築すればよい。

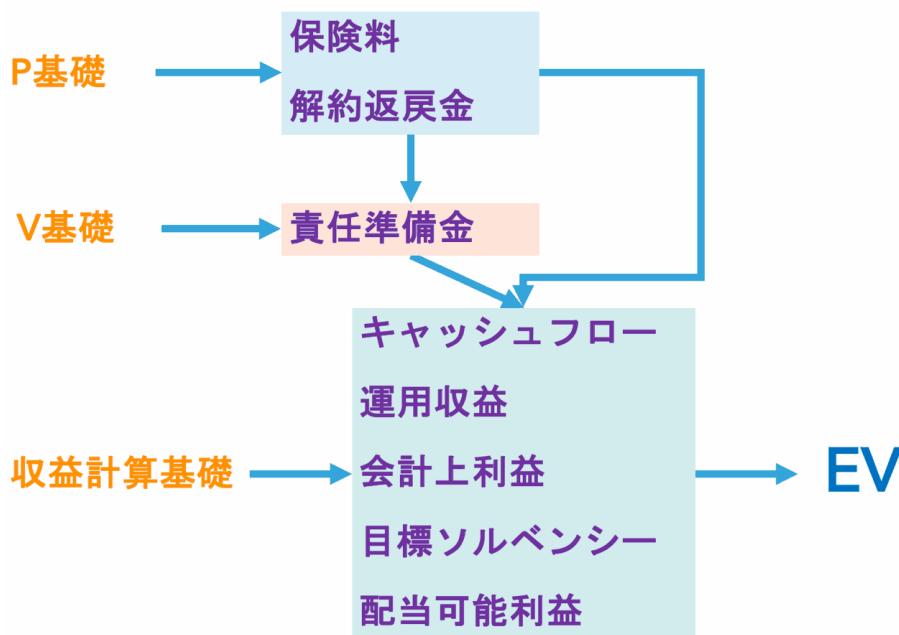


図27: 各種基礎率から潜在価値への計算パイプライン

- ・本節では、この自動微分機能を用いた**保険商品の収益性のアサンプション詳細感応度の自動導出**について紹介する。バックプロパゲーションを感応度計算に利用することである。本来ならば別途の論文とすべきところかもしれないが、この実装そのものは本研究を通して得られたことと、独立した論文とするほどでもないと考え、本論文に収録することとした。本節の実装もGitHubにアップしている。
- ・フレームワークはPyTorchを用いた。特段高度な機能は用いていないが、累積和・累積積を高速に計算可能なcumsum・cumprodのあるフレームワークが望ましいと考えられる( $\text{I}_{\times}$ 等の計算を高速に行うため)。

### ■終身保険の潜在価値感応度

#### ◇数式

<sup>39</sup> <https://www.tensorflow.org>

<sup>40</sup> <https://pytorch.org>

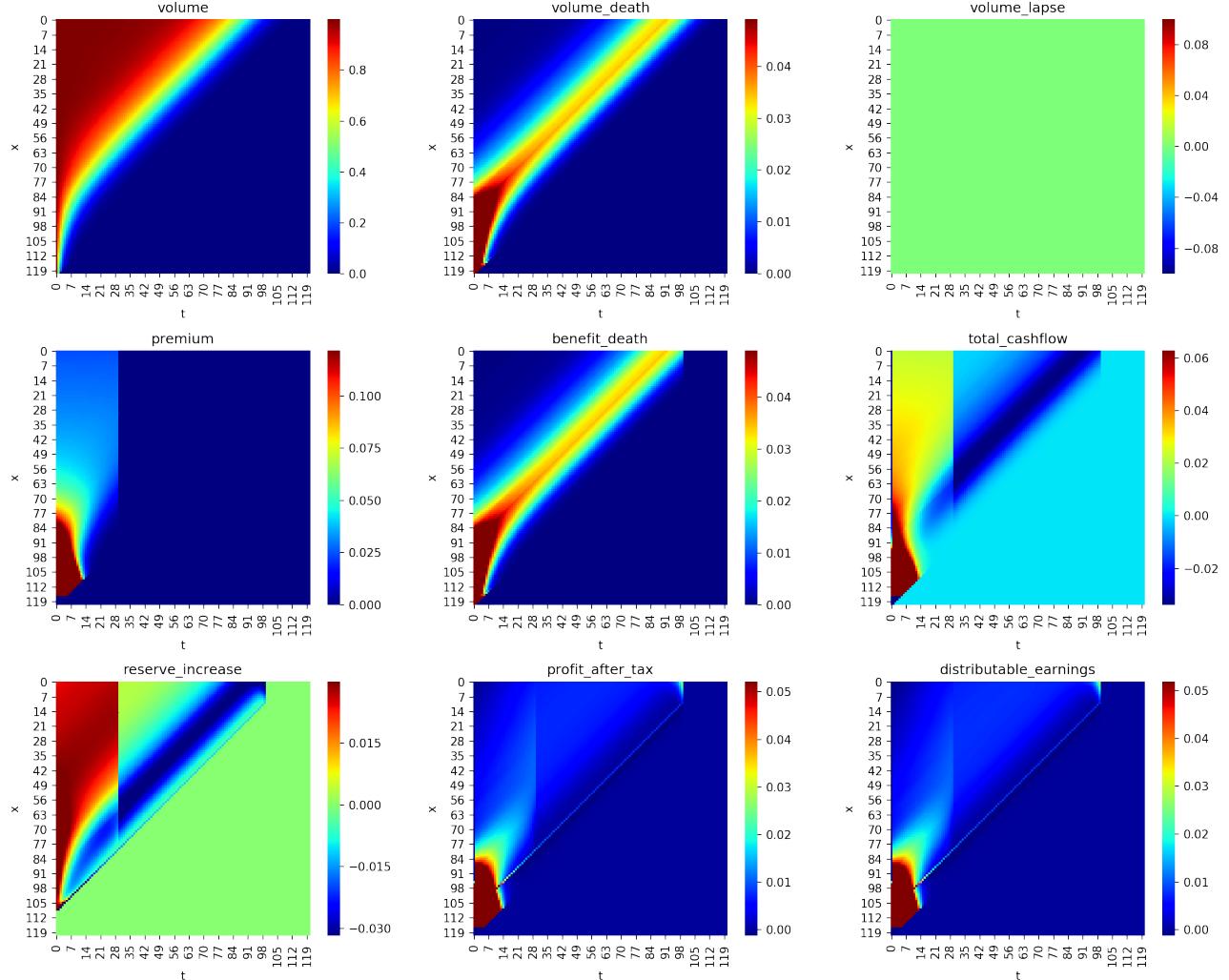
<sup>41</sup> <https://chainer.org>

<sup>42</sup> ニューラルアクチュアリアルモデルとでも呼びたくなるが、別に学習能力を持っているわけではないし、少なくとも本論文ではそのような側面については考へない。あくまで流用に過ぎないので誤解しないように気をつけたい。

- ・保険料や責任準備金の計算・EV の計算は極めて標準的なものを用いたので、詳細は実装を参照して頂きたい (EV 計算はかなり簡略化している) .

## ◇結果

- ・図 28 は契約年齢毎 (X 軸) における各経過年数 (Y 軸) での各項目の値を示したヒートマップである。EV のベースとなる配当可能利益は右下のセグメントである。保険商品としては保険期間 100 年の 30 年払の定期保険 (概ね終身保険) とした。幾つかのグラフで横軸で 30 のあたりで払込満了による境界線が現れていることが分かる<sup>43</sup>。



**図 28:** 契約年齢  $x$ ・経過年数  $t$  における EV 計算上の各項目 : X 軸が経過年数  $t$ , Y 軸が契約年齢  $x$  である。まず上段が左から保有 (volume) ・死亡数 (volume\_death) ・解約数 (volume\_lapse) である。中段が保険料収入 (premium) ・死亡保険金支払 (benefit\_death) ・キャッシュフロー (total\_cashflow) である (今回は簡単のため解約率はゼロとしている)。下段が責任準備金繰入 (reserve\_increase) ・税引後利益 (profit\_after\_tax) ・配当可能利益 (distributable\_earnings) である。保険料収入は 30 年目までであり、更に高齢の方が保険料・死亡保険金支払いが短期に集中することが表現されている。

- ・図 29 は各年齢に保険金額 1 の保有がある場合の EV に対する各種の死亡率 (pricing : P 基礎・reserve : V 基礎・valuation : 収益性計算基礎) の年齢別感応度 ( $\partial \text{EV} / \partial q_x$ ) である。P 基礎を上げると単純に保険料が増大するため EV は増大し、V 基礎を上げると若年齢部分ではネガティブリザーブ気味になることからやはり EV は増大し、収益性計算基礎を上げると保険金支払いが増えるため EV は低下することを示している。しかし、実務上よく行われるリスクシナリオ (死亡率一律 20% 上昇など) のように全年齢で V 基礎死亡率を上昇させると、各歳の感応度をシ

<sup>43</sup> 簡単のため、P 基礎死亡率としては第 3 分野用標準生命表・V 基礎としては死亡保険用・収益性検証用としては年金開始後用を用いた。また、予定事業費などはかなり適当に設定している。

ヨック量で加重平均したものとなるため最終的には V 基礎の上昇は EV を減少させる効果となることが確認できる。

・本論文では簡単のため死亡率の偏導関数しか求めていないが、同様にして死亡数  $d_x$  に対する勾配を求めた上で、死亡数を、死亡率を発生率・保有契約数を回数とする二項分布を仮定して、その標準偏差を各歳別に求めてかけあわせることで、リスク量に相当するものも導出できる。あるいはベイズ的に考えて基礎率の不確実性に相当する標準偏差を各種発生率の勾配にかけあわせることでも基礎率の不確実性のリスク量に相当するものも導出できる。

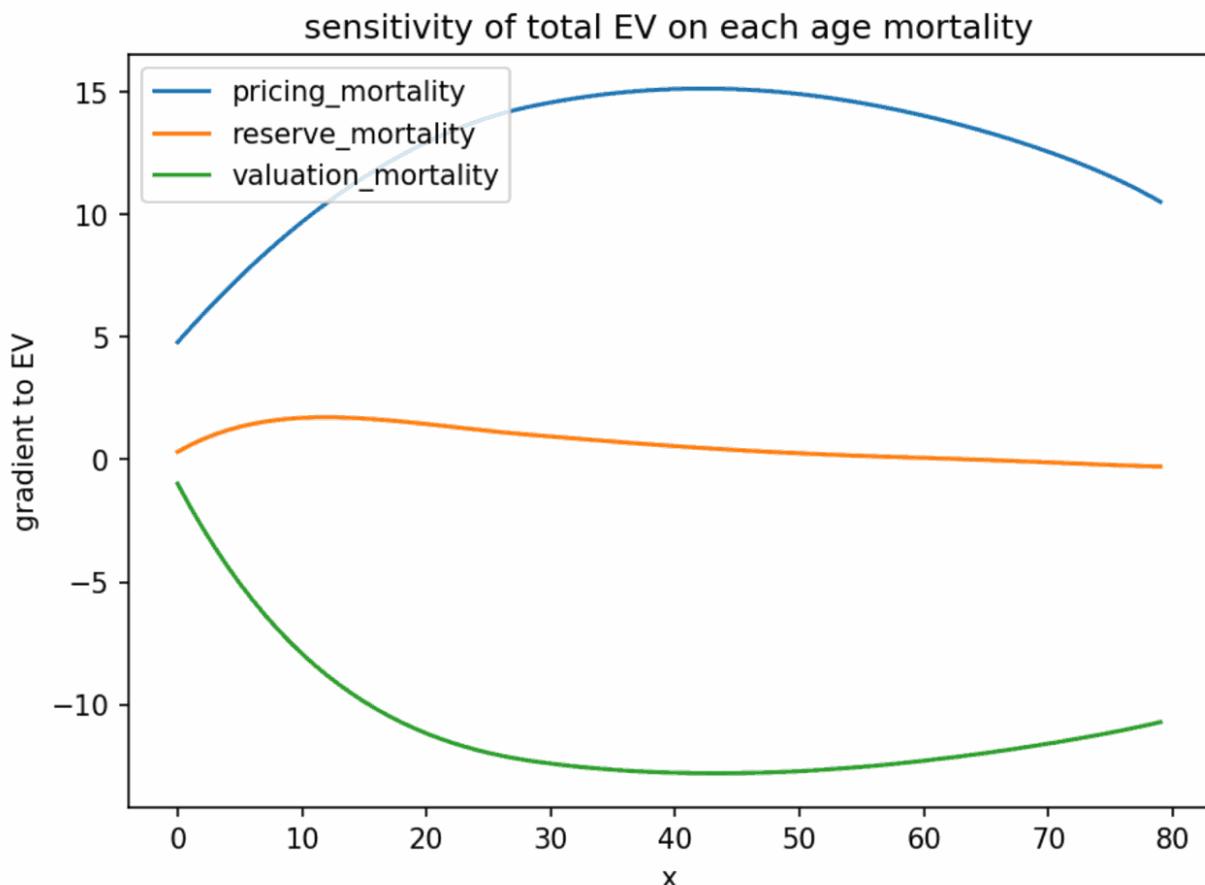


図 29: 会社全体の EV の死亡率各歳に対する感応度  $\partial \text{EV} / \partial q_x$  : 詳細な設定は本文に述べたとおりである。P 基礎を上げると保険料が単純に増大するためプラス・V 基礎を上げると若年齢ではネガティブリザーブになるためプラス・収益性検証用を上げると保険金支払が増えるのでマイナスとなっている。

・図 30 は各歳別の EV の感応度 ( $\partial \text{EV}_x / \partial q_{x+t}$ ) である。上段は単純な偏導関数、下段は偏導関数を元の死亡率で割ったもの ( $\partial \text{EV}_x / \partial q_{x+t} / q_{x+t}$ ) である。

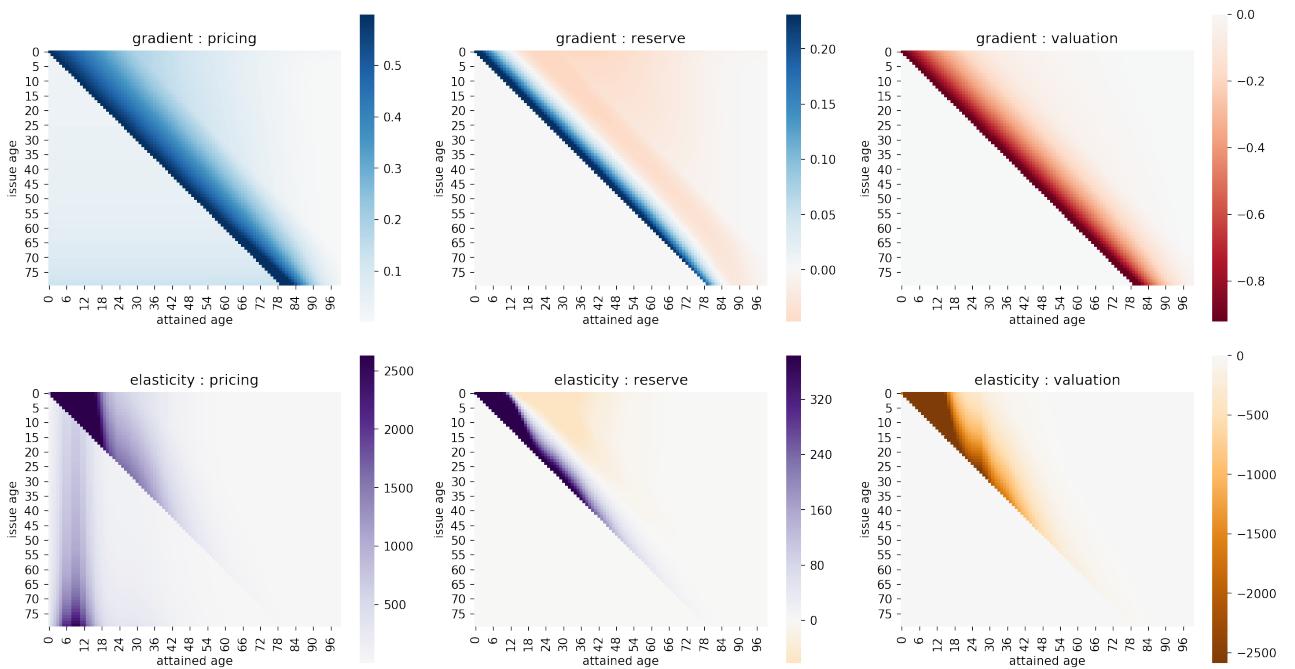


図 30: 各歳別の EV の死亡率感応度  $\partial \text{EV}_x / \partial q_{x+t}$  (上段),  $\partial \text{EV}_x / \partial q_{x+t} / q_{x+t}$  (下段)

・尚、何れも計算時間はさほどかからない (MacBookPro 15inch Early 2013 にて 100 件で数秒のオーダー) ので、メモリに気をつけつつミニバッチ計算を行う等の工夫をすれば、現実的な時間内で会社の保有全体に対して行うことも可能だろう。

## ■変額年金の責任準備金

・次に、確率的な計算に対する例として、**変額年金の最低保証責任準備金の、動的解約率を構成する各種パラメータに対する感応度**を求めてみる。本来、本節で述べた方法は確率論的な計算にはそのままでは適用できないが、VAE で用いられる reparametrization trick を用いることで変額年金などのモンテカルロ計算も含めた大概のケースに対処可能となる。本項では簡単な GMDB タイプのみ扱うが、GMWB・定期引出・ラチェットタイプも扱えるだろう。

### ◇数式

・数式は以下の通りである。年単位計算・動的解約率のパラメータは保険期間を通じて同じにするなど、かなり簡略化している。また、動的解約率はシグモイド関数を用いたユニットプライスの関数として表現している。シグモイド関数とは図 31 に示すような関数 ( $1/(1+e^{-x})$ ) で、概ね  $\pm 5$  で  $0 \sim 1$  に収束する関数で機械学習ではお馴染みのものである。

$$\text{unit price}_{t+1} \sim N(1 + \mu - \text{cog}, \sigma) \times \text{unit price}_t$$

$$\text{lapse} = \text{lapse}_{\min} + (\text{lapse}_{\max} - \text{lapse}_{\min}) \times \text{sigmoid}\left(5 \times \frac{\text{unit price} - \text{position}_{\min}}{\text{position}_{\max} - \text{position}_{\min}}\right)$$

- ・  $\mu$  : 期待收益率 - 予定事業費率 ( $= -1.5\%$ )
- ・  $\text{cog}$  : 最低保証保険料 ( $= 0.5\%$ )
- ・  $\sigma$  : ボラティリティ ( $= 20\%$ )
- ・  $\text{lapse}_{\min}$  : 解約率の最小値 ( $= 0\%$ )
- ・  $\text{lapse}_{\max}$  : 解約率の最大値 ( $= 30\%$ )
- ・  $\text{position}_{\min}$  : 解約率が最小となるユニットプライス ( $= 80\%$ )
- ・  $\text{position}_{\max}$  : 解約率が最大となるユニットプライス ( $= 140\%$ )
- ・ その他、死亡率は第三分野標準生命表・割引率は  $1.5\%$ ・保険期間は 10 年とした。

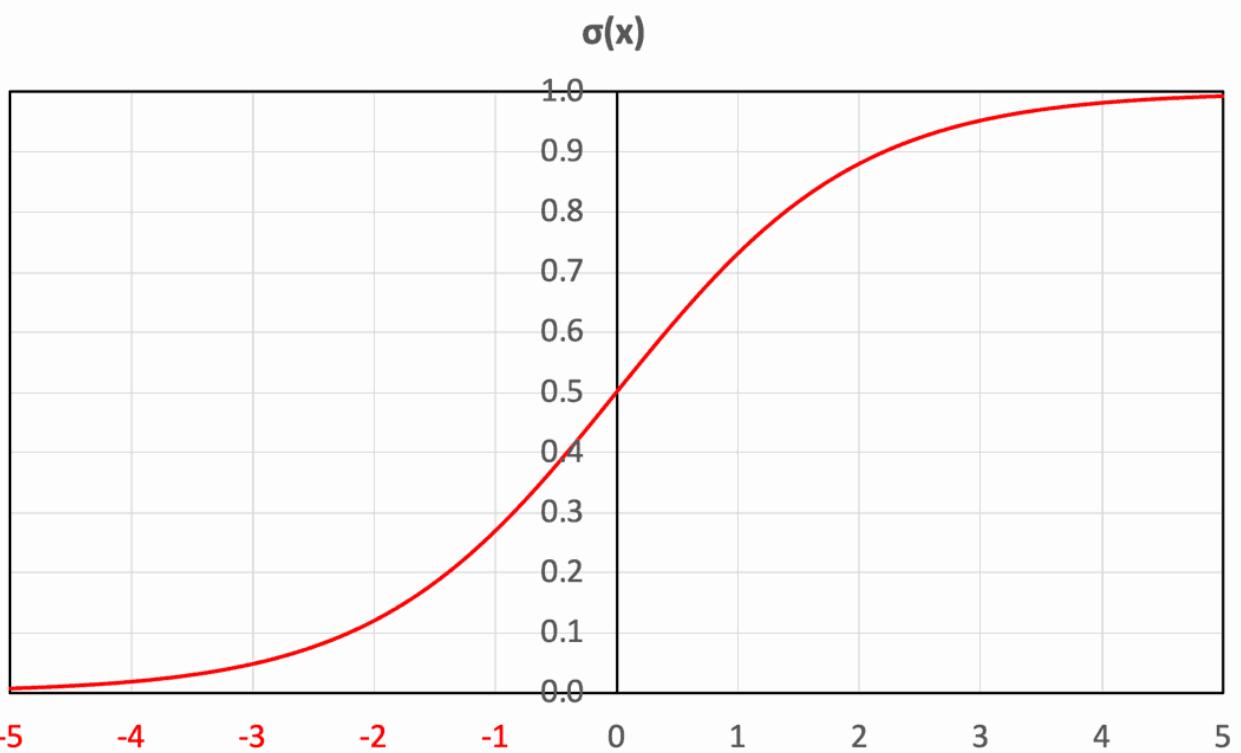


図 31: シグモイド関数

◇結果

- ・図 32 は契約時点の各ユニットプライスにおける責任準備金を構成要素毎に示したものである。ユニットプライスが低いほど保険料収入現価部分が小さくなるのに対し、死亡保険金（の最低保証部分）が大きくなり、（最低保証）責任準備金が大きくなることが分かる。

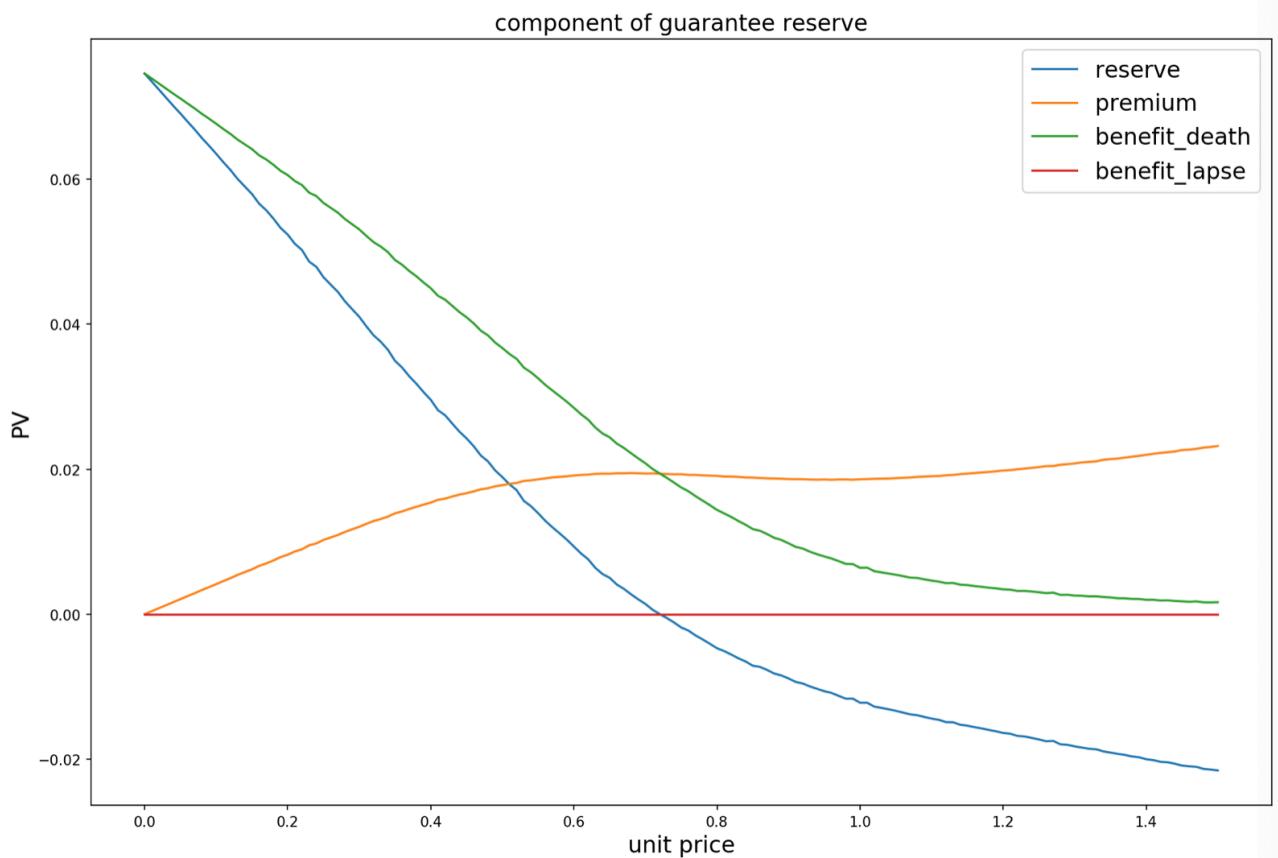


図 32: 最低保証責任準備金の構成要素：X 軸がユニットプライス（積立金）・Y 軸が各構成要素の大きさである。ユニットプライスが低いと最低保障責任準備金やそのベースとなる死亡保険金支払原価（benefit\_death）が増大していることが分かる（ここでは最低保証部分のみ表示している）。

- ・図 33 は同じく各ユニットプライスにおける、責任準備金に対する各パラメータの感応度（勾配）を示したものである。ファンド成長率を示す  $\mu$  は常にマイナスである。ボラティリティを示す  $\sigma$  はユニットプライスに応じて若干向きが異なる。

- ・このように、動的解約率の責任準備金に対する影響は、ユニットプライスに応じて複雑に変動することが分かる。

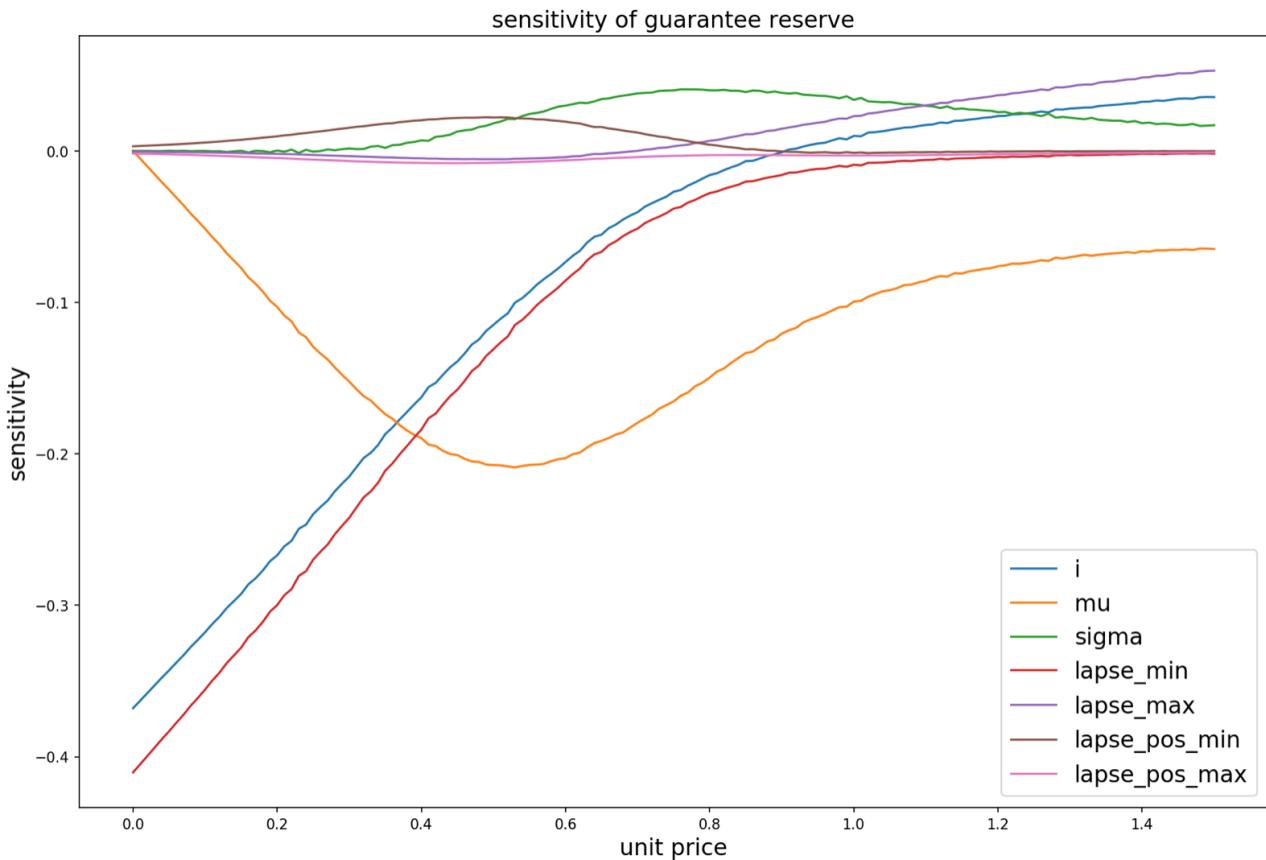


図 33: 最低保証責任準備金の各種パラメータに対する感応度 ( $\partial V / \partial i$ , etc.) :  $i$  が予定利率・ $\mu$  がファンド成長率・ $\sigma$  がファンド成長率のボティリティ・ $\text{lapse}_{\min}$  は動的解約率の最小値・ $\text{lapse}_{\max}$  は最大値・ $\text{lapse}_{\text{pos},\min}$  は動的解約率が概ね最小値になるユニットプライス・ $\text{lapse}_{\text{pos},\max}$  は最大値になるユニットプライスを表している。ファンド成長率を示す  $\mu$  は一貫して責任準備金を低下させるためマイナスである。動的解約の影響は一定の傾向はあるが比較的まちまちで複雑である。

## ■ 雜記

- ・PyTorch で自動微分機能を有効にする場合は、途中の配列のどの要素でもエラーが発生していくではない。このことから、配列のパディング・ゼロ除算の回避が随所で必要であり、端の面倒な処理が必要である。本論文の実装では一旦最終年齢に等しい長さのパッドを追加した後に最終年齢までの長さで全体をスライスする方法でやや雑に処理している。
- ・デフォルトの 32 ビット浮動小数だと計算基底の誤差がそれなりに貯まるようである。場合によっては 64 ビットを用いても良いかもしれない。
- ・ソルベンシーマージンのリスク計算時にルートを取り必要があるが、ルートは変数がゼロだと勾配が発散するため、ゼロにならないように微少な数でフロアする必要がある。
- ・今回は途中経過に関しては勾配を保存していないが、計算途中で生じる数 ( $A_x$  とか  $D_x$  とか) の勾配も計算することが可能である。
- ・保険料・責任準備金・収益性指標の導出過程はそのまま分析プロセスに直結する。数理設計は保険料などはどう導出されるべきかあるいはどうリスクが拾えているかという観点のみならず、どう分析されるかという観点からも設計されうるし、それが可能になるということである。これは生成モデル的な観点に近いかもしれない。
- ・本節で紹介した方法は一次導関数に相当するものだから当然精度には問題がある。ディープラーニングフレームワークの中にはより高次の偏導関数が算出できるものもあるが、そのレベルになってくると基礎率全ての勾配を出しても手に負えないと考えられるので、何らかの情報の簡潔化が必要だろう（何次オーダーまで出せばどれだけ精度があるか、あるいは複合シナリオでどういった影響があるかといった実験は勿論可能だろうが、ある程度の基礎率のまとめを行わないと不毛になりそうである）。どう簡潔化するかは基礎率の関係をどうモデル化するかでもあるので、分析者の洞察が問われることになる。

## ■B.付録：破産事象の事後分布の導出について

### ■概要

- ・2019年現在、メジャーなディープラーニングフレームワークは確率論的プログラミングの機能を備えるようになってきている（TensorFlow Probability<sup>44</sup>・Pyro<sup>45</sup>）。
- ・このため、前節で作成したようなモデルに少し手を加えて確率論的なモデルへと転換することが可能である<sup>46</sup>。勿論、確率論的なモデルに転換しただけでは将来事象や潜在価値の確率分布が得られるだけであり意外性は低い。しかし、ここにベイズ統計を組み込むことで、会社の破産事象を観測値として与えることによって「何が生じて会社が破産したか」を事後分布として導出することが理屈の上では可能になる<sup>47</sup>。本節では極めて簡単なモデルで試した結果を紹介する。
- ・フレームワークはPyroを用いた。PyroはPyTorchをバックエンドとする確率論的プログラミングのフレームワークであり、Uberから2017年より提供されているものである。Pyroは基本的には変分推論を前提に設計されていると考えられるが、本論文では問題スケールが小さいことと事後分布の形状に関する知見が無いことからMCMC・NUTSを用いることにした。

### ■全般

- ・時間的な制約があったため、本節では前節のようなモデル構築は行わず、10年定期保険についてのみ扱うこととする。また、簡単のため保険料・事業費・金利・責任準備金・ハードルレートについては考えない。つまり、保険金支払いのみを考えるものとする。死亡数は本来二項分布に従うと考えられるが、二項分布でモデルすると計算時間が非常に長くなるため、正規近似で扱うこととした。また、死亡によって残存数は減少していくが、死亡率が低く期間も短いため、この効果は無視することとした。
- ・結果を分かりやすくするために、保険金のパターンは以下の図34に示す非常に極端なケースに限定した。また、死亡率は簡単のため、0.1%～1%へと10年で線形に上昇するものとし、開始時の人数は1000人とした。このため、保険金総額は正規分布の線形和であるので正規分布となる。

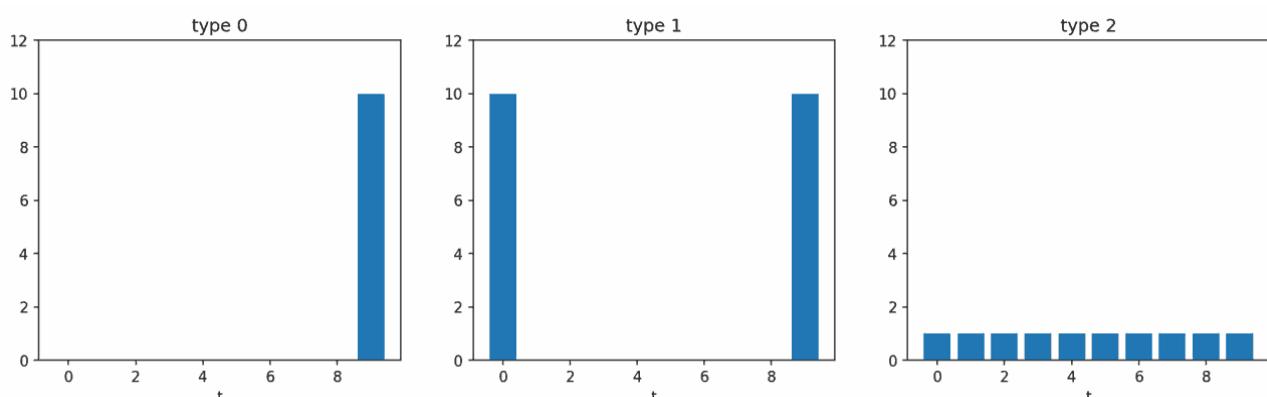


図34: 保険金のパターン：最後に10・最初と最後に10・常に1の3パターン。

<sup>44</sup> <https://www.tensorflow.org/probability/>

<sup>45</sup> <http://pyro.ai>

<sup>46</sup> 基礎率から確率変数へと変換する部分だけ書き換えなければならない。

<sup>47</sup> これと通常のショックシナリオとはかなり意味合いが異なる。例えば、ワンタイムで30%死亡でも2年に分けて15%・15%死亡でも結果にはほぼ変わりないだろうが、後者がショックシナリオとして利用されることには恐らく無い。しかし、純粋に確率論的には後者の方が確率が高く、本節の方法も基本的には後者の分布を出力する。これはショックシナリオの本分は解釈可能性・対策考慮などにあるのであって、単純にショックを与えて結果の数値だけを見て終わりにするといった単純計算にあるのではないからである。そういう意味では、本節の方法は劇的でない緩やかなストレスシナリオを生成する傾向にあるため、安全割増の水準への適正の方が高いかもしれない。

## ■結果

### ◇潜在価値の分布

- ・潜在価値（ただの死亡保険金）の分布は図 35 の通りである。全て正規分布になっている。

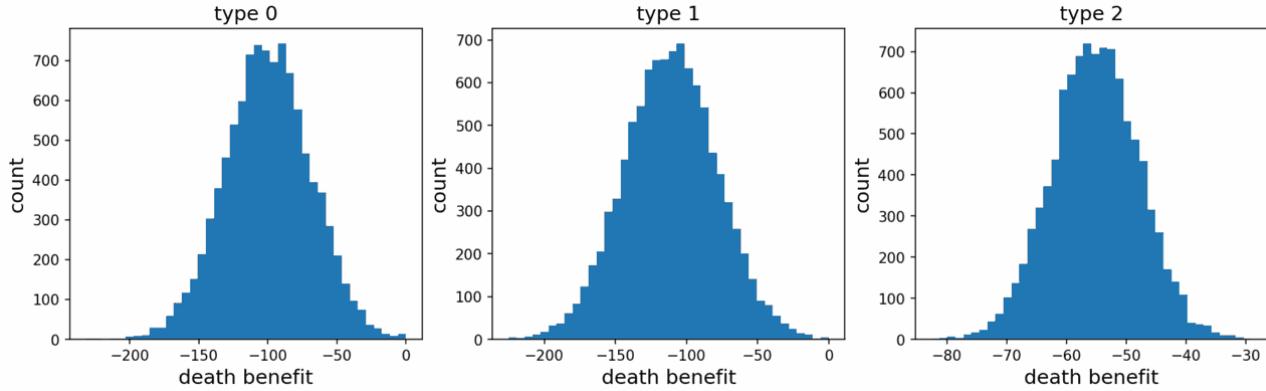


図 35: 潜在価値の分布

### ◇死亡数の事後分布

- ・破産としては集合 $\{EV < 0\}$ にすべきだが、今回はとりあえず  $EV = \text{保険金期待値} - 100$  を観測値として与えた上で、死亡数の事後分布を見ることにした。各経過年数における死亡数の事後分布は各々以下の図 36 の通りである。Y 軸が各経過期間、X 軸がサンプル番号（10 おき）のヒートマップとなっている。サンプル数は 10000 とし、10 個おきにマップしている。先ず type0 では保険金の発生する  $t=9$  に死亡が集中していることが分かる。また、全期間で保険金が同額になっている type2 では概ね死亡率に比例した形で死亡が発生している。これら 2 つについては概ね予想通りの結果が得られていると言えそうである。しかし、type1 では  $t=0$  と  $t=9$  に死亡が集中し、かつ  $t=9$  の方が多めになることが想定されるが、実際にはそうなっておらず、おかしな挙動になっている。これは問題が意外に難しいのか、MCMC サンプラーの設定がよくないのかは残念ながら詰め切れなかった。

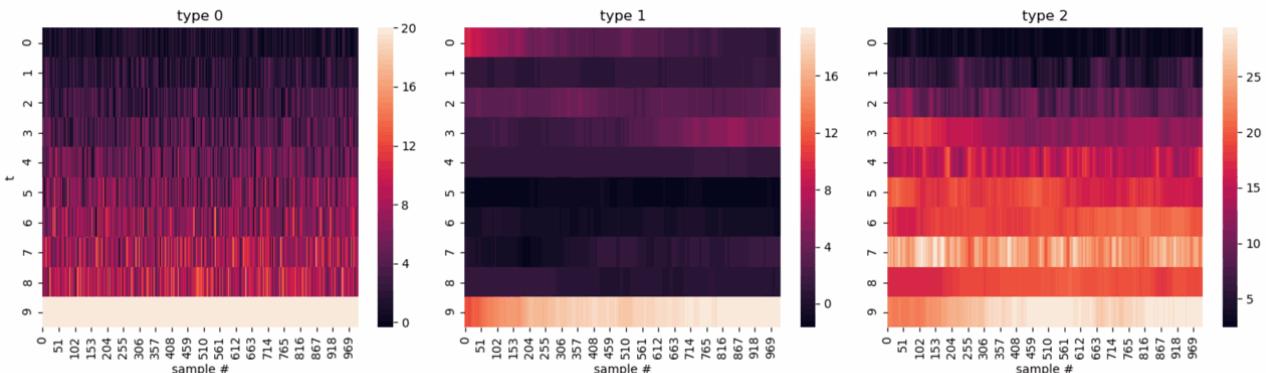


図 36: 各保険金タイプにおける死亡数の事後分布：type 0 と type 2 は概ね想定通りだが、type 1 の挙動は納得感がない。

・結果の解釈には注意が必要である。この結果は死亡数が二項分布（を近似した正規分布）に従うならば、このような死亡数分布に概ねなっているだろう、と言っているに過ぎないからである。先ず、既知の死亡率による二項分布を仮定している点で基礎率設定ミス・アンダーライティングリスク・逆選択などは一切考えていない。これらの影響まで考えたい場合は、死亡率自体をアサンプション確信度に応じて確率分布にして階層ベイズ化する必要がある。更に事後分布を検証することでどちらのリスクが大きく発現しているかを確認する必要がある。

・この方法は前節の方法と比べると確率を直接扱っているという点ではより直接的・基礎論的だが、計算時間は莫大で勾配のような比較的解釈・利用しやすい結果をもたらすものでもない（おまけに type 1 のような不可解な現象も発生しているし、事後分布のヒストグラムを見てもあまり綺麗ではなかった）。単に所定の  $EV$  になるような状態を求めたいだけならば、前節の方法で  $EV$  の値を指定した上で死亡数などを学習パラメータ化して教師あり学習を用いるか（そういう解は

当然複数あるので何らかの正則化のようなものが必要だと思われる），あるいは尤度を求めて勾配上昇法で最尤推定を行う方が早いと思われる。多峰性分布が疑われるケース，分布幅が必要なケース，柔軟な拘束条件が必要なケースなどに限って利用するか，詳細な検証や考察を目的とした基本要素レベルでのモデルケースでの使用に限るなどするのがよいかもしれない。

以上

## 参考文献

1. Volodymyr MnihKavukcuoglu, David SilverKoray. Human-level control through deep reinforcement learning. : Nature, 2015.
2. David SilverHuang, Chris J. MaddisonAja. Mastering the game of Go with deep neural networks and tree search. : Nature, 2016.
3. Elena Krasheninnikova, Javier García, Roberto Maestre, Fernando Fernández. *Reinforcement learning for pricing strategy optimization in the insurance industry*. : Engineering Applications of Artificial Intelligence, 2019.
4. Brian D. ZiebartMaas, J.Andrew Bagnell, and Anind K. DeyAndrew. Maximum Entropy Inverse Reinforcement Learning. : Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008.
5. AmirRamachandran and EyalDeepak. Bayesian Inverse Reinforcement Learning. : IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence, 2007.
6. alexirpan. Deep Reinforcement Learning Doesn't Work Yet. (オンライン) 2018 年.  
[https://www.alexirpan.com/2018/02/14/rl-hard.html?utm\\_campaign=News&utm\\_medium=Community&utm\\_source=DataCamp.com](https://www.alexirpan.com/2018/02/14/rl-hard.html?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com).
7. Peter HendersonIslam, Philip Bachman and etc.Riashat. Deep Reinforcement Learning that Matters. : arXiv, 2017. 1709.06560.
8. Yuhuai WuMansimov, Shun Liao, Alec Radford and John SchulmanElman. OpenAI Baselines: ACKTR & A2C. : Open AI, 2017.
9. Timothy P. LillicrapJ. Hunt, Alexander Pritzel and etc.Jonathan. Continuous control with deep reinforcement learning. : arXiv, 2015. 1509.02971.
10. Tim SalimansHo, Xi Chen and etc.Jonathan. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. : arXiv, 2017. 1703.03864.
11. Felipe Petroski SuchMadhavan, Edoardo Conti and etc.Vashisht. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. : arXiv, 2017. 1712.06567.
12. Shixiang GuLillicrap, Ilya Sutskever and Sergey LevineTimothy. Continuous Deep Q-Learning with Model-based Acceleration. : arXiv, 2016. 1603.00748.
13. Xue Bin PengKumar, Grace Zhang and Sergey LevineAviral. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. 出版地不明 : arXiv, 2019. 1910.00177.
14. BartoS. Sutton and Andrew G.Richard. Reinforcement Learning: An Introduction. : A Bradford Book, 2018. 0262039249.
15. 哲郎森村. 強化学習. : 講談社, 2019. 4065155916.
16. OpenAI Spinning Up. (オンライン) OpenAI, 2018 年.  
<https://spinningup.openai.com/en/latest/>.
17. 隆宏久保. 機械学習スタートアップシリーズ Python で学ぶ強化学習 [改訂第 2 版] 入門から実践まで. 出版地不明 : 講談社, 2019. 4065172519.
18. LapanMaxim. Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more. : Packt Publishing, 2018. 1788834240.
19. えるエル. 強化学習における好奇心. (オンライン) 2019 年.  
[https://speakerdeck.com/learn\\_learning3/qiang-hua-xue-xi-niokeruhao-qi-xin](https://speakerdeck.com/learn_learning3/qiang-hua-xue-xi-niokeruhao-qi-xin).
20. 力矢高橋. 高精度人工知能は帰納と演繹をまとめて「サボる」. (オンライン) SmartNews, 2017 年. <https://developer.smartnews.com/blog/2017/03/machine-learning-game-theory/>.