強化学習を用いた推薦システム

17311282 大友一馬

2021 年 8 月 7 日

1　はじめに

情報技術の発展に伴い，動画共有サイトや e-commerce service などが社会に浸透している．それに伴い様々な形式の情報が爆発的に増加している．そのため，膨大な情報量から，個人が望む情報の獲得が困難となっている．したがって，個人が望む情報を推薦する推薦システムの確立が様々な Web サービスにおいて急務になっている．

推薦システムは，主に教師あり学習の枠組み[*1]でおこなわれてきた．つまり，個人が望む情報を表すフィードバックに基づいて，深層学習（機械学習）を最適化し，それに合致した情報の推薦を行う．しかしながら，教師あり学習であるが故に，従来の最適化には以下の疑問が残る．

- フィードバックは全データに対して極めて少ないデータ数 (スパース) である．つまり，短絡的なフィードバックに最適化しても真に望む情報の獲得は難しい．
- 短絡的なフィードバックのみに着目し，ユーザの長期的な満足度の最適化は行えない．

この問題に対処するために，強化学習に基づいて推薦を行う手法が提案されている．強化学習は教師あり学習に比べ，報酬を定めれば必ずしもフィードバックに基づく最適化は行う必要がなく，また報酬の定義を定めれば，長期的な満足度を最適化できる．

2　目的

本レポートの目的は，上記で述べた強化学習に基づいた推薦システムを理解することである．本レポートの構成は以下の通りである．

1. はじめに，従来の教師なし学習や教師あり学習に比べて，強化学習はどのような違い（立ち位置）があるのかを説明する．
2. 次に，迷路問題に焦点を当て，強化学習がどのように学習するかを最も初歩的であるモデルベースに基づいて説明する．
3. 2. で説明したモデルベースでは，現実問題をうまく網羅できない．モデルベースでは，環境の定式化が必要となるため，環境の定式化が難しい場合には適用が難しい．その対策として，モデルフリーの強化学習を説明する．

[*1] 細かく言えば，クリックだけで最適化を行うランキング学習が主流である．

4. 最後に，強化学習の推薦システムへの適用を説明する．Policy Gradient に基づく強化学習に基づき，新たに Importance weights を導入しているためそれを説明する．そして，大規模なデータの取り扱いと top-K の情報の推薦を行う際の工夫にも触れる．

本レポートは「機械学習スタートアップシリーズ Python で学ぶ強化学習 入門から実践まで」を参考にまとめている．ただし，個人的解釈も含めているため，もしかしたら間違っている場合があるかもしれないが，その際はご指摘いただけると幸いである．

3　強化学習の立ち位置

2012 年に画像分類のコンペティションである ImageNet Large Scale Visual Recognition Competition (ILSVRC) において，深層学習モデルである AlexNet が大差をつけて優勝して以来，深層学習は人間の認識精度を超えるまでに発展している．深層学習はそのモデルの汎用性の高さより，画像認識だけでなく物体認識や，音響信号処理，自然言語処理やゲーム AI にまで応用が広がっている．

表 1　学習の方法による違い

	教師なし学習	教師あり学習	強化学習
用途	クラスタリング，分散表現等	画像分類，テキスト分類等	ゲーム AI，自動ロボット等
備考	人間のラベル付けが必要ない 一般的に，教師あり学習に比べて精度は劣る	人間のラベル付けが必要 強化学習に比べて，複数の行動による結果を考えられない	複数の行動の結果を学習可能 学習が不安定

深層学習 (機械学習) は大まかに学習の方法によって三つに分けられる．表 1 に各々の用途と備考をまとめる．教師なし学習は主に表現学習（特徴抽出）の枠組みで用いられる．例えば，文脈を考慮した単語特徴抽出（分散表現）を行う際は，Wikipedia などの文を用いて，入力単語とその単語前後の予測問題を学習させる．その時，あえて $\mathbb{R}^n$ の特徴ベクトル空間に射影する線形変換を施す．すると，その特徴ベクトル空間においては，入力単語とその周辺単語を表す特徴ベクトルが近くなる，すなわち，関連するように学習する．このように特徴ベクトルに変換を施すことにより，一次元ベクトルで扱うことが可能となり，様々な機械学習処理において文章や画像等を扱いやすくすることを可能としている．その他にも，画像の特徴や構造に着目して類似していたらまとめるクラスタリングも教師なし学習の一種として挙げられる．

教師あり学習は主に分類問題を行う際に用いられる学習の方法である．例えば，画像に何が写っているかを分類する画像分類問題においては，ある画像とそれは何かを表す人間が付与したラベル（猫や犬など）が一対一で対応するデータを収集する．そして，深層学習（機械学習）に画像が入力されたら，それに対応するラベルを出力するように学習を行う．

強化学習は，行動を行い，その行動に対して環境から定義される報酬の最大化を行う．例としては迷路探索などが挙げられる．報酬をゴールまでの移動ステップ数の最小化として考えると，その複数の行動を予測し，満足する行動を予測する．深層学習を用いた強化学習は深層強化学習と言われ，行動の判断を深層学習が行う．

教師あり学習は一般に教師あり学習に比べてラベル付けを行う必要はないため，コストは低いが精度は教師あり学習に比べて劣る場合が多い．教師あり学習はラベルに基づいて，学習を行う．しかしながら，複数の行動から得られる結果への最適化はむずかしい．それに対応して，強化学習は複数の行動から得られる結果への

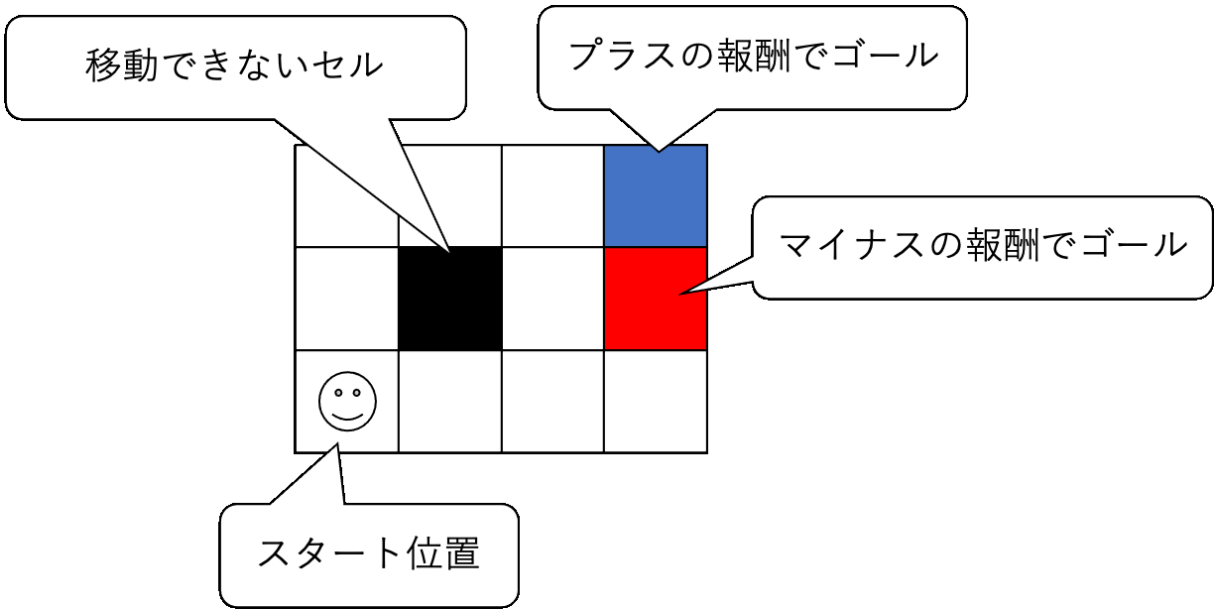

図 1　迷路問題の環境

最適化を可能とする．すなわち，教師あり学習に比べて長期的な行動予測が可能である．この性質は，後に説明する推薦システムにおいて大きく活かされている．しかしながら，学習の安定性は教師あり学習に比べて少ない[*2]．

本章では，強化学習の立ち位置をまとめた．強化学習は教師あり学習に比べて，複数の行動から得られた結果への最適化を行う．この性質は，（報酬を定義できれば）機械学習の長期的な予測を可能とする．

4　モデルベース強化学習

4.1　迷路問題

最初に図 1 のような迷路探索を考える．強化学習の前提として，与えられた環境が一定のルールに従っていることを仮定する．数学的に言えば，**マルコフ性 (移動後の行動状態は，移動前の行動状態のみに依存する)** に従っていることを仮定する．そのマルコフ性に従うことを，マルコフ決定過程と呼ぶ．マルコフ決定過程（Markov Decision Process: MDP）の構成要素は以下の 4 つとなる．

- s: 状態 (State) 迷路問題においては，マス目の位置
- a: 行動 (Action) 迷路問題においては，上下左右の移動方向
- T: 状態遷移の確率 (Transition function) 状態と行動を引数に，遷移先（次の状態）と遷移確立を出力する関数 すなわち，次に移動するマスの確率を表す．
- R: 即時報酬 (Reward function) 状態と遷移先を引数に，報酬を出力する関数

[*2] 教師あり学習はラベル数に対応して精度が向上するというスケーラビリティが存在する．しかしながら，強化学習にはそれがなく，初期値によっては解が収束しない場合もある．これは，再現性ということで議論されているが，本レポートでは内容が逸脱するので触れない．もし教師あり学習に帰着できる場合は，教師あり学習を行う方が良い．

青のマスにゴールすれば $+1$ の即時報酬が得られる．しかしながら，赤のマスにゴールすれば -1 の即時報酬を得てしまう．ただし，白いマスは移動するごとに -0.2 の即時報酬を得てしまう（移動が多くても報酬が少なくなることを表現）．強化学習がやりたいこととしては，上下左右に移動していかに報酬を高くゴールすることである．

4.2 報酬

では，報酬はどのように考えれば良いか？右や上に移動して得られる即時報酬の総和を行えば良い．最終的な行動 T で終わる場合，時刻 t における即時報酬 r の総和 G_t は以下のように定義できる．

$$G_t \stackrel{\text{def}}{=} r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T,$$

しかしながら，この式には未来の即時報酬が含まれるため，行動終了まで計算ができない．ただ，強化学習側としては，報酬を高くゴールしたいため，報酬時刻 t で即時報酬の総和を知っておきたい．そこで，未来の即時報酬は見積もりを立てることでなんとかする．しかしながら，見積もりは不確かな値となるため，値を割り引く．この割り引くための係数を割引率 (Discount factor)γ と呼ぶ．すなわち，

$$G_t \stackrel{\text{def}}{=} r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T = \sum_{k=0}^{T} \gamma^k r_{t+k+1},$$

割引率 γ は 0 から 1 までの値を取り，未来の値になるとより割り引かれることがわかる．このように未来に得られる値を割り引いて計算した値を**割引現在価値**と呼ぶ．

しかしながら，上式には依然として以下の二つの問題が残存する．

- r_{t+1}, r_{t+2} など，未来の即時報酬の値が判明していること
- 必ず未来の値が得られるということ

一つ目は，式を再帰式に置き換えることで解決ができる．具体的には，

$$\begin{aligned} G_t &\stackrel{\text{def}}{=} r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T, \\ &\stackrel{\text{def}}{=} r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots + \gamma^{T-t-2} r_T), \\ &\stackrel{\text{def}}{=} r_{t+1} + \gamma G_{t+1}, \end{aligned} \tag{1}$$

すなわち，値 G_{t+1} に置き換えて，それを適当な値に置き換えることで報酬の算出が可能となる．

二つ目は，即時報酬を期待値と見なすことで解決できる．つまり，行動確率を定義できれば，行動確率とその即時報酬の積と，その全ての行動の総和で表すことができ，未来の値を大まかに予測することができる．この時，行動を定義する方法が二種類存在する．

- エージェントは保持している戦略に基づき行動する．
- エージェントは常に「価値」が最大になる行動を選択する．

戦略 π に基づく場合，行動 a をとる確率は $\pi(a|s)$ と表すことができる．ただし，s は状態（迷路問題で言えば位置情報）を表す．そして，遷移先 s' へは遷移関数から導かれる確率 $T(s'|s,a)$ で遷移する．戦略 π に基づいて行動した結果得られる報酬を $V_\pi(s)$ とすると，この $V_\pi(s)$ は式 (1) 同様に

$$V_\pi(s_t) = E_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})],$$

と再起的に定義できる．ここで E は期待値を表現し，下つき文字はその条件下であることを表す．さらに，報酬 r を報酬関数 $R(s', s)$ に置き換え，より変数を細かく表現すると

$$V_\pi(s_t) = \sum_a \pi(a|s) \sum_{s'} T(s'|s, a)(R(s', s) + \gamma V_\pi(s')), \tag{2}$$

と表される[*3]．この再帰的かつ期待値で表現する式を **Bellman Equation（ベルマン方程式）**と呼ぶ．

式 (2) は戦略 π を用いるため，戦略ベースの報酬であった．では，「価値」が最大になる行動を選択するにはどのような報酬の定義が良いか？戦略 π に乗っ取らず，行動 a に基づいて常に期待値（報酬）の最大値を考えれば良い．すなわち，

$$V_\pi(s_t) = \max_a \sum_{s'} T(s'|s, a)(R(s', s) + \gamma V_\pi(s')), \tag{3}$$

この式 (2),(3) の違いは，Policy ベースと Value ベースとしてよく区別される．

迷路探索問題に関しては，現在の状態から報酬を算出可能であるので，あらかじめ期待値の式より変形しておく．

$$V_\pi(s) = R(s) + \gamma \max_a \sum_{s'} T(s'|s, a)V(s'), \tag{4}$$

4.3 動的計画法による報酬予測

式 (4) では，報酬が最大であることを求める必要がある．しかしながら，$V(s')$ は，状態数が複雑になればなるほど全てをあらかじめ計算することが困難となる．それに対処するために，動的計画法 (Dynamic Programming: DP) が用いられる．動的計画法は，初めは適当な値を設定しておき，複数回計算を繰り返すことで値の精度を上げていく方法である．

最初に Value ベースで考える．Value に基づいて繰返し反復することを Value Iteration と呼ぶ．計算回数を i で表すと，以下のように書くことができる．

$$V_{i+1}(s) \stackrel{\text{def}}{=} \max_a \{\sum_{s'} T(s'|s, a)(R(s', s) + \gamma V_i(s'))\}, \tag{5}$$

ここで重要なのが，$i+1$ 回目の計算には，i 回目の計算結果を用いるということである．これは，メモ化と呼ばれる動作である．これにより，予測される報酬 $V(s)$ を正確な値に近づけていく．

次に Policy ベースを考える．Policy に基づいて繰返し反復することを Policy Iteration と呼ぶ．戦略 π に基づく行動 a への確率の積で実現できる．

$$V_{i+1,\pi}(s) \stackrel{\text{def}}{=} \sum_a \{\pi(a|s) \sum_{s'} T(s'|s, a)(R(s', s) + \gamma V_{i,\pi}(s'))\}, \tag{6}$$

Value Iteration と比較すると，行動確率も含めた期待値で表現される．つまり，戦略の確率に基づいて報酬予測が行われることが式からわかる．収束条件としては，$|V_i(s) - V_{i+1}(s)|$ がある閾値以下になったらある程度予測される報酬が正確な値になったとして終了する場合が多い．動的計画法で求めた次の報酬に基づいて，報酬が最大となる行動を予測可能となる．

[*3] 行動の確率とそこから次の状態に遷移する確率は条件付き確率の積で表すことができる．サイコロを 2 個順番に振り，1 回目に出る目と 2 回目に出る目のペアの確率と同じである．ここでは r_{t+1} の報酬を表現したいので，$t+1$ 時間の考えられる全ての行動と状態を考え，期待値を出す．

5　モデルフリー強化学習

4 章では，モデルベースによる強化学習を説明した．ここで，強化学習は行動を試していないことに留意したい．言い換えれば，これは行動前の計画を立てていると言える．これは遷移関数 T と報酬関数 R が明らかな場合にのみ可能である．現実には，遷移関数と報酬関数が不明，もしくは定式化できないという場合が多くある[*4]．

モデルフリー強化学習は，その遷移関数と報酬関数が不明な場合でも学習を可能とする．しかしながら，モデルベース強化学習に比べて以下のことに留意する必要がある．

1. モデルベース強化学習に比べて，遷移関数が不明であるが故に，環境の探索が必要となる．しかしながら，探索にはコストがかかる場合が多くあり，また探索に固執すると報酬の最大化が行えない．
2. 報酬関数が不明であるが故に，どのように報酬を最大化する行動への最適化を行うかに複数方法がある．

次節では，上記の問題に対しての対処法を説明する．

5.1　探索と活用

1 つ目に関しては，探索と活用（報酬の最大化）をバランスよく行う Epsilon-Greedy 法で対処可能である．設定された Epsilon の値に基づいて探索と活用のどちらかを行う．具体的には，0.2 と Epsilon を設定したら，20% の割合で探索を行い，80% の割合で活用を行う．

表と裏が出る確率がバラバラなコインが何枚か存在するとして，表が出たときに報酬が得られるゲームを考える．この場合，なるだけ表が出やすいコインを「探索」して，「活用」してそのコインをたくさん投げること望ましい．このような問題は，**多腕バンディット問題（Multi-armed bandit problem）**と呼ばれる．コイントスで表を足すように Epsilon-Greedy 法で探索，活用した結果を図 2 に示す．ここで，縦軸が報酬，横軸が試行回数である．まず，Epsilon が 0，すなわち探索を全く行わない場合，報酬は著しく低い．一方で，Epsilon が 0.8，0.5 の場合であっても，活用が全く行われずに報酬の獲得を行えていない．Epsilon が 0.1, 0.2 の場合，試行回数とともに報酬が増加していることがわかる．一般的な Epsilon の設定としては，0.1 前後がよく使われる設定である．注意して欲しいのは，これはコイントスに限った話ではなく，強化学習でも環境の探索とその活用に Epsilon-Greedy 法が用いられる．

5.2　報酬の最大化方法 -実績か予想か？-

報酬関数が未知なため，**実績から行動の修正を行う**か**予測から行動の修正を行う**かが考えられる．実績から行動の修正を行う方法は，主に**モンテカルロ法 (Monte Carlo Methods)** があげられる．また，予測から行動の修正を行う場合は，**TD 法 (Temporal Difference Learning)** があげられる．またその中間として，**Multi-step Learning** と **TD(λ) 法**が存在する．

[*4] 教習所の座学で，出発前に地図を見てルートを計画するということを紹介されたが，これはまさにモデルベースであると言える．地図が手元にない場合，実際に運転して場当たり的に移動するのがモデルフリーであると言える．

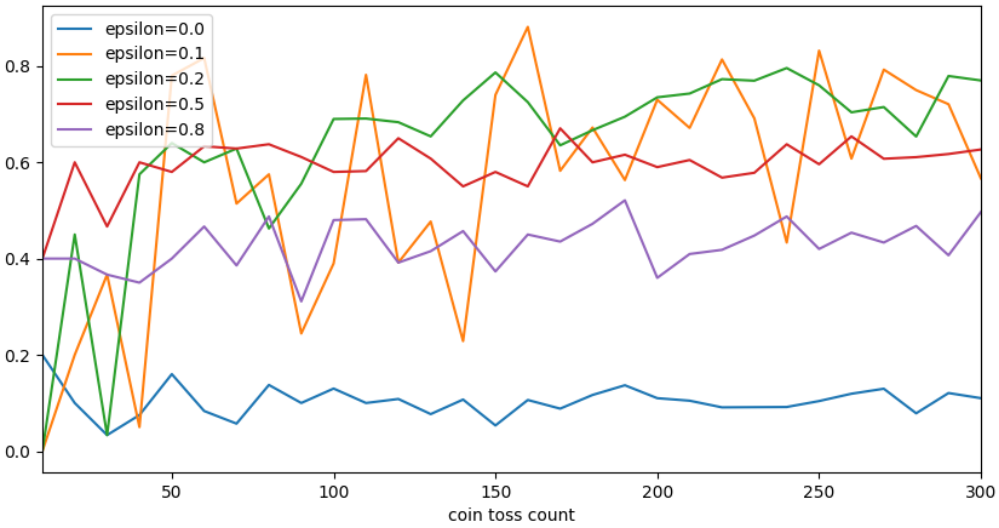

図 2　Epsilon-Greedy 法による表が出やすいコインの探索

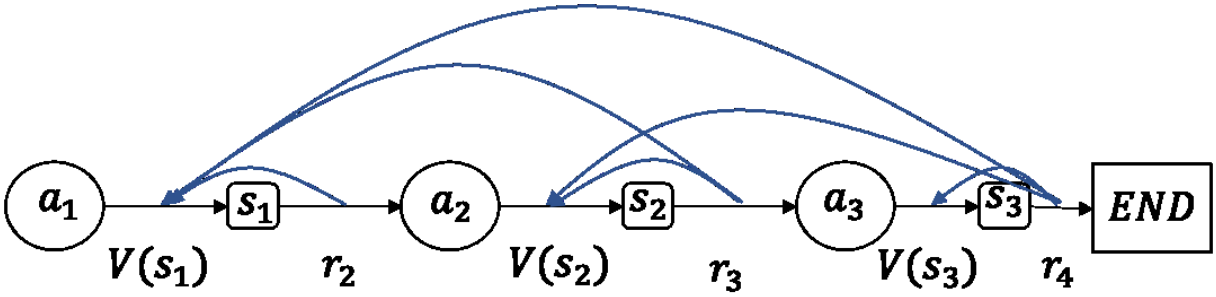

図 3　モンテカルロ法による行動の修正．予測される報酬を最後に得られる報酬から最適化する．

5.2.1　実績から行動の修正を行う方法

図 3 にモンテカルロ法の概要を示す．モンテカルロ法は，最後に実際に得られた報酬から，予測される報酬 $V(s_t)$ の修正を行う．青矢印がその修正を示している．式で表すと

$$V(s_t) \leftarrow V(s_t) + \alpha((r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T) - V(s_t)). \tag{7}$$

5.2.2　予測から行動の修正を行う方法

図 4 に TD 法の概要を示す．TD 法は，TD 法はステップごとに予測される報酬 $V(s_t)$ の修正を行う．青矢印がその修正を示している．式で表すと

$$V(s_t) \leftarrow V(s_t) + \alpha((r_{t+1} + \gamma V(s_{t+1}) - V(s_t)). \tag{8}$$

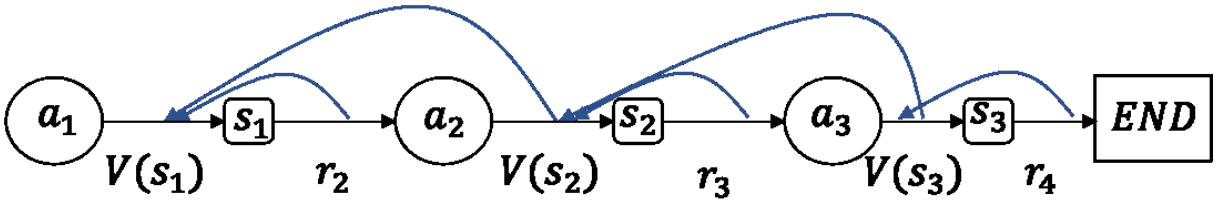

図 4　TD 法による行動の修正．予測される報酬を最後に得られる報酬から最適化する．

表 2　実績・予測のメリットデメリット

	実績	予測
メリット	実際に得られた報酬から修正するため， 修正方針は正しいと言える	行動終了を待たずに修正が可能
デメリット	行動終了を待たなければ 修正が行えず遅い	得られた報酬以外に，予測された報酬も含めて 修正するため，修正方針が正しくない場合もある

5.2.3　ハイブリッド方式

上記で説明した実績からと予測からの行動の修正には，それぞれ表 2 のメリットとデメリットがある．これらのデメリットを補うために，業績と予測を組み合わせた行動修正が行われている．その一つである Multi-step Learning は，式 (7) に基づいて行動終了まで見ていた報酬を，2,3 ステップなど，ユーザが指定したステップまでの報酬のみで学習を行う．また，TD(λ) 法は各ステップの報酬を λ で合成する．各ステップの報酬は以下のように計算できる．

$$
\begin{aligned}
G_t^1 &= r_{t+1} + \gamma V(s_{t+1}), \\
G_t^2 &= r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}), \\
&\vdots \\
G_t^T &= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_T.
\end{aligned}
$$

合成式は以下のようになる．

$$
G_t^\lambda = (1-\lambda) \sum_{n=1}^{T} \lambda^{n-1} G_t^{(n)} \tag{9}
$$

λ はユーザが指定する 0 から 1 までの値だが，ここで $\lambda = 0$ の場合を考えると，直前の報酬が 1，それ以外の値は 0 掛けされることがわかる．すなわち，直前の値のみを報酬として考えるため，TD 法と同じであることがわかる．一方で，λ が 1 に近くなると，行動終了の結果を重視するようになり，モンテカルロ法に近づく[*5]．

5.3　TD 学習を例にした Value ベースと Policy ベースの比較

モデルベースと同様に，行動学習の考えに Value ベースと Policy ベースが存在する．この節では，TD 学習に注目して Value ベースと Policy ベースの学習方法について説明する．Value ベースの学習は Q-learning

[*5] 総和の計算式からわかるように，これは行動終了まで計算を待たなければならない．この待たなければならない TD(λ) 法を Forward-TD(λ) 法といい，待たなくてもよい方法は Backward-TD(λ) として提案されている．

と呼ばれ，更新式は以下のようになる．

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma \max_{a_{t+1}}(V(s_{t+1})) - V(s_t)) \tag{10}$$

(4) に示す TD 学習の更新式と同じであるが，予測される報酬が最大となる行動を選択する．Policy ベースの学習は SARSA（State-Action-Reward-State-Action）と呼ばれ，更新式は以下のようになる．

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma(V(s_{t+1})) - V(s_t)) \tag{11}$$

ただし，a_{t+1} の行動は戦略 π から算出されるとする．

また，その他の手法として Actor-Critic や Policy gradient などが存在する．後述する推薦システムにおいては，Policy gradient の一種である REINFORCE を用いている．

5.4 Deep Q-Network

最後に，囲碁で人間に勝利した Alpha GO に用いられている Deep Q-Network について軽く触れる．Q-learning の更新式 (10) では，内部で $V(s)$ をテーブル化して参照していた．この V を Q 値，そしてテーブル化されたものを Q テーブルと呼ぶ．しかしながら，行動や状態が新たに定義される問題を解こうとすると，このテーブルの定義が難しくなることがわかる．また，行動や状態が連続値で表現される場合，テーブルそのものを定義することはできない．そこで，深層学習を用いて Q 値を表現することが試みられた．これが Deep Q-Network である．

迷路探索においては，上下左右の行動報酬の表現が必要であるが，深層学習の線形層で上下左右の行動価値を表すことは簡単に行える．価値ベクトル $\boldsymbol{y}$，位置ベクトル $\boldsymbol{x}$ と深層学習のパラメータ $\boldsymbol{W}$,$\boldsymbol{b}$ とした時，

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{12}$$

と表すことができる．ここで，深層学習のパラメータ $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ を適切に表すことができれば，価値 $\boldsymbol{y}$ の予測が可能となる．ではどのように適切なパラメータを表せば良いか？ Deep Q Network では Q-learning に乗っ取り，

$$L = E[\frac{1}{2}(r_{t+1} + \gamma \max_{a_{t+1}}(\boldsymbol{y}_{t+1,i-1}) - \boldsymbol{y}_{t,i})^2] \tag{13}$$

で表すことができる．ここで，$\boldsymbol{y}_{t,i}$ は時刻 t における学習 i 番目のパラメータから出力される予測報酬を表している．これを確率的勾配効果法で最適化することで，パラメータ θ の適切な表現が可能となる．

深層学習で表すことで，テーブルの定義が不必要になるだけでなく，様々な形式の入力を受け付けることを可能とする．例えば，ボールキャッチゲームのスコアを最大化する問題においては，画面全てを畳み込みニューラルネットワークで入力することが可能となる．また，現実問題において，音響特徴が有効である場合にはそれも入力として受け付けることが可能となる．

しかしながら，単純に最適化するだけでは，学習が不安定となる問題がある．Deep Q Learning では，以下の三つのテクニックを駆使して学習を安定化させている．

Experience Replay
: 行動履歴は時系列に得られるため，相関があらわれてしまう．そこで，一旦行動履歴を格納して，そこからランダムサンプリングで学習する行動履歴を取り出す方法.

Clipping
: 報酬を全行動を通じて成功は 1, 失敗は-1 と統一する方法.

Fixed Target Q-Network

一定期間固定されたパラメータから報酬を算出する方法

また，Deep Q-Network をさらに改良させた Rainbow という手法も提案されている．

6 強化学習を用いた推薦システム

1 章で述べた通り，強化学習で推薦システムを構築した手法が提案されている．本章では，その具体的な手法を説明する．

6.1 記号の定義

- $\mathcal{S}$: ユーザの連続的な状態遷移空間
- $\mathcal{A}$: 推薦に使用可能なアイテムを含む，離散的な行動空間
- $\boldsymbol{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ で表される遷移確立
- R: $\mathcal{R} \times \mathcal{A} \to \mathbb{R}$ で表される報酬関数．そして $r(s, a)$ で状態と行動に対応する報酬と表現する．
- ρ_0: 初期状態分布
- γ: 割引率

6.2 強化学習の定式化 -Policy gradient-

やりたいこととしては，ユーザの状態 $s \in \mathcal{S}$ からアイテムの推薦 $a \in \mathcal{A}$（行動）を行いたい．すなわち戦略 $\pi(a|s)$ を最適化したい．そこで，期待される報酬が最大化されるようにする．つまり，

$$\max_{\pi} E_{\tau \sim \pi}[R(\tau)], \text{where } R(\tau) = \sum_{t=0}^{|\tau|} r(s_t, a_t). \tag{14}$$

ここで，$\tau = \{s_0, a_0, s_1, \cdots\}$ の行動遷移，ただし，$s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \boldsymbol{P}(\cdot|s_t, a_t)$ である．

REINFORCE の手法に乗っ取り，戦略勾配は log トリックに基づいて

$$E_{\tau \sim \pi}[R(\tau) \nabla \log \pi_\theta(\tau)] \tag{15}$$

と表される．オンラインで学習をする際には，戦略勾配は学習中の戦略に基づいて算出された行動遷移によって算出されるため，戦略勾配の推論は普遍であり以下のように分解できる，

$$\sum_{\tau \sim \pi_\theta} R(\tau) \nabla_\theta \log \pi_\theta(\tau) \approx \sum_{\tau \sim \pi_\theta} [\sum_{t=0}^{|\tau|} R_t \nabla_\theta \log \pi_\theta(a_t|s_t)] \tag{16}$$

ただし，割引率を適用して $R_t = \sum_{t'=t}^{|\tau|} \gamma^{t'-t} r(s_{t'}, a_{t'})$ としている．

6.3 Off-policy 補正

得られるデータは，過去の戦略によって得られたデータしか得られないため，現在の戦略を評価することは難しい．そこで，On-policy から Off-policy へ補正することを考える．挙動戦略 β を用いて，現在の戦略 π_θ

の評価を行う．

$$\sum_{\tau\sim\beta}\frac{\pi_\theta(\tau)}{\beta(\tau)}[\sum_{t=0}^{|\tau|}R_t\nabla_\theta\log(\pi_\theta(a_t|s_t))], \tag{17}$$

ここで，

$$\frac{\pi_\theta(\tau)}{\beta(\tau)}=\frac{\rho(s_0)\prod_{t=0}^{|\tau|}\boldsymbol{P}(s_{t+1}|s_t,a_t)\pi(a_t|s_t)}{\rho(s_0)\prod_{t=0}^{|\tau|}\boldsymbol{P}(s_{t+1}|s_t,a_t)\beta(a_t|s_t)}=\prod_{t=0}^{|\tau|}\frac{\pi(a_t|s_t)}{\beta(a_t|s_t)} \tag{18}$$

これを適用すれば policy のバイアスを抑えた推論が可能となるが，バリアンスが増大になるという問題がある．そのために，$|\tau|$ から時刻 t までを計算するようにし，一次近似を行う．

$$\prod_{t=0}^{|\tau|}\frac{\pi(a_{t'}|s_{t'})}{\beta(a_{t'}|s_{t'})}\approx\prod_{t=0}^{t}\frac{\pi(a_{t'}|s_{t'})}{\beta(a_{t'}|s_{t'})}=\frac{P_{\pi_\theta}(s_t)}{P_\beta(s_t)}\frac{\pi(a_t|s_t)}{\beta(a_t|s_t)}\approx\frac{\pi(a_t|s_t)}{\beta(a_t|s_t)} \tag{19}$$

これを用いて，バリアンスを抑えた推論が可能となる．

$$\sum_{\tau\sim\beta}[\sum_{t=0}^{|\tau|}\frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)}R_t\nabla_\theta\log(\pi_\theta(a_t|s_t))], \tag{20}$$

6.4 戦略 π_θ の深層学習化

深層学習で戦略を算出することを考える．ユーザベクトル $\boldsymbol{u}_{a_t}\in\mathbb{R}^m$，状態ベクトル $\boldsymbol{s}_t\in\mathbb{R}^n$ を入力として，次の状態の遷移確率 $\boldsymbol{P}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}$ を RNN で予測する．論文では，Chaos Free RNN (CFN) が最も安定して計算コストが良いとして用いている．状態の更新は以下のように行われる．

$$\boldsymbol{s}_{t+1}=\boldsymbol{z}_t\odot\tanh(\boldsymbol{s}_t)+\boldsymbol{i}_t\odot\tanh(\boldsymbol{W}_a\boldsymbol{u}_{a_t}) \tag{21}$$

$$\boldsymbol{z}_t=\sigma(\boldsymbol{U}_z\boldsymbol{s}_t+\boldsymbol{W}_z\boldsymbol{u}_{a_t}+\boldsymbol{b}_z) \tag{22}$$

$$\boldsymbol{i}_t=\sigma(\boldsymbol{U}_i\boldsymbol{s}_t+\boldsymbol{W}_i\boldsymbol{u}_{a_t}+\boldsymbol{b}_i) \tag{23}$$

ここで，$\boldsymbol{z}_t$,$\boldsymbol{i}_t\in\mathrm{R}^n$ はアップデートとゲートを表している．ユーザの状態 $\boldsymbol{s}$ に応じて，戦略 $\pi_\theta(a|\boldsymbol{s})$ は温度付き Softmax よりモデル化される．

$$\pi_\theta(a|\boldsymbol{s})=\frac{\exp(\boldsymbol{s}^{\mathrm{T}}\boldsymbol{v}_a/T)}{\sum_{a'\in\mathcal{A}}\exp(\boldsymbol{s}^{\mathrm{T}}\boldsymbol{v}_{a'}/T)} \tag{24}$$

ここで，$\boldsymbol{v}_a\in\mathrm{R}^n$ は行動 $\boldsymbol{a}\in\mathcal{A}$ を表現する埋め込みである．

挙動戦略 β の推論は，ネットワークは上記と共有し過去の状態と行動のペアを用いて Softmax で推論する．ただし，勾配情報が流れないようにしている．

6.5 Top-K 推薦への対応

推薦では，アイテムを複数種類提示する問題が一般的である．つまり，今までは Top-1 推薦問題を考えていたが，行動は複数種類考える必要がある．そこで，行動を $a\to A$，戦略を $\pi\to\Pi$ に置き換えて考える．しかしながら，単純に複数行動 A に拡張すると計算量が膨大になる．論文では対策として，重複も加味した A' に置き換えて考えている．つまり，

$$\Pi_\Theta(A'|s)=\prod_{a\in A'}\pi_\theta(a|s) \tag{25}$$

A' を算出後，重複したアイテムを除去する．

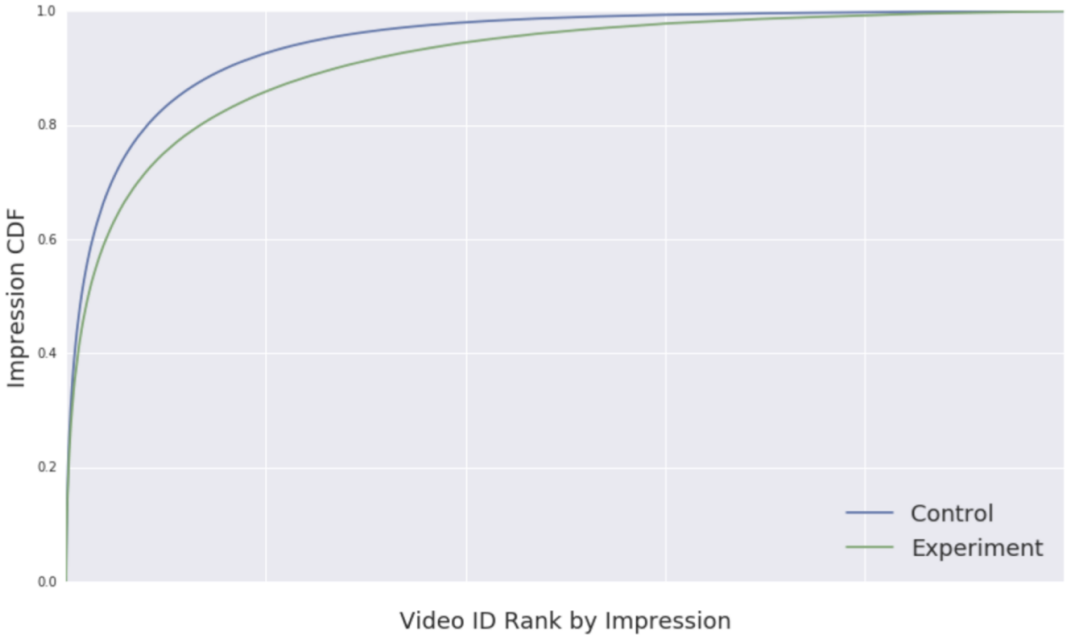

Figure 4: CDF of videos nominated in control and test population according to rank of videos in the control population. Standard off-policy correction addresses the “rich get richer“ phenomenon.

図 5　インプレッション (サムネイルが出現した回数) の累積関数

6.6　結果

論文の筆者らは Google 所属であり，実際に Youtube の環境を用いた実験をしている．Off-policy 補正を行わない強化学習と補正を行う強化学習用いた推薦システムで A/B テストを行い，その有効性を評価している．評価尺度は推薦アイテムの多様性である．図 5 にインプレッション（サムネイルが出現した回数）の累積関数を示す．青が control(Off-policy バイアス補正を行わない手法) と緑が補正を行なった手法である．青の方が左上に分布が偏っていることがわかる．すなわち，人気なアイテムを推薦する傾向が緑線より高い．対して補正を行なった手法は，アイテムの推薦の多様性が向上したことがわかる．

6.7　所感

強化学習で推薦を行う手法は全く新しく，また環境の定義も難しそうである印象である．特にこの論文では，Youtube で A/B テストを行なっているため，我々には再現ができない．しかしながら，将来的には必ず流行る手法だと思っているため，その動向を注視していきたい．