強化学習の基礎

講　　師：小池 康晴・鮫島 和行
レポーター：根本 憲・浦久保 秀俊

2001 年 4 月 6 日 金曜日

1　はじめに

川人さんが提案されたフィードバック誤差学習などの中にはフィードバックコントローラというものが常に入っています。私が強化学習を始めたきっかけは、このフィードバックコントローラがどのように学習されるのかが知りたいというものでした。昨日、Barto の論文中に Actor - Critic の話が出てきましたが、ここで Actor はまさしくフィードバックコントローラの役割をしており、それが何らかの報酬を与えることで学習ができるのです。そこで、この Actor - Critic の強化学習を使ってフィードバックコントローラを学習するようなものを作ろうと思い、強化学習の研究を始めました。私は当時、たまたま自動車をつくっている会社にいたので、現在も車を運転するモデルに関して研究をしています。入力情報として道路と車の間のずれや速度等を入れ、また報酬として道の上にのっていればゼロ、道から外れれば罰として－１を与えるといった簡単なものでも、いろいろな道幅を走らせればちゃんと学習して速度に応じて安全に道路を走るといった行動ができるようになります。初日の銅谷さんのビデオでロボットが立っていたような話もありますし、今では強化学習がそういったいろいろな制御に応用できる可能性があることが徐々に分かってきています。

2　強化学習とは？

2.1　教師あり学習との違い

強化学習は ”環境との相互作用による学習”であると言えます。実際何か行動してみて (例えばポインティングであればポインティングしてみて)、ちゃんとしたターゲットについていたら報酬（サルだったらジュース）を与えるというように、自分から環境にはたらきかけて環境から報酬をもらえるといった相互作用により学習していくというものです。また、ターゲットに向かっていくというようなものを使って学習していくということから、“ ゴール指向型の学習 ”とも言えます。この強化学習が教師あり学習と一番違うところは、どうやって手を伸ばすか、筋肉を活性化させるかという教師信号は全く与えず、ターゲットまで手を伸ばせ、といったように行為に対して報酬を与えることで学習できるという点であり、そこが面白いところではないかと思います。

テキストの最初の方に赤ちゃんの話しが入っています。（赤ちゃんが寝返りしたビデオを示しながら）この赤ちゃんは生後半年くらいであり、寝返りができるようになって２・３日目ですが、赤ちゃんが寝返りしたり立って歩くようになるのは、別に何をしろと教えているというわけではありません。これが強化学習であるかどうかは判りませんが、教師あり学習ではありません。どの赤ちゃんでも寝返りすることから、遺伝的に組みこまれていると考える人もいるかもしれません、し

かし、赤ちゃんの体格は生まれてから半年で倍くらいになりますし、それからも徐々に成長していくことを考えると、人間の腕や足は脳にとっては環境と同じ外界のものです。そのため、それを動かしながら学習するということと併せて、強化学習に近いものが使われていると考えられます。

次に、教師あり学習と強化学習との違いについてです、教師あり学習は正しい行動というものがどういうものかが明確に示されているのに対し、強化学習というのは正しい行動というのは与えられず、試行錯誤を通じて環境に適応していきます。教師あり学習は、ターゲットと実際の出力の差をとる、フィードバック誤差学習だと言えます。一方強化学習は、出力に対してなにか評価（報酬とか罰）を与え、将来的に報酬が最も多くなるよう学習していくことでシステムを良くしていきます。

2.2 制御との関係

（Barto の論文の図 11.A,B を入れる）

強化学習を制御の視点から考えると、（図 11.A）の様になります。図中、Controller は、Context とフィードバック信号との差を入力として、出力をおこないます。川人さんのフィードバック誤差学習の話にもあるように、Controller が良いものでなければ、フィードバック誤差学習で制御対象の逆モデルを学習しようと思ってもいい逆モデルができません。このようなシステムにおいて、Controller をどのように設計するのか私にとってはわかりませんでした。次に、先程の Controller を Actor にかえます（図 11.B)。昨日の論文にあるように、この Actor-Critic という形式だと、環境（Controlled System）からの出力を Context を使って評価して評価信号というものを作り、それを用いて Critic と Actor 両方を学習していくといった関係になります。ですからフィードバック誤差学習の良いところは実際に運動しながら学習できるところにあり、順逆モデルの様に順モデルをまず学習しておいてそれを使って逆モデルを学習しようといった順番はなくて、両方同時進行で、オンラインでどどん学習できます。順モデル、逆モデル、フィードバックコントローラが学習しながら獲得できるのです。

また、目標軌道の入力方法も問題になります。車の例であれは、車に乗っていると道路は見えます。それが Actor にきて、今どこにいるのかがわかります。それだけで、目標軌道がなくても道から外れないように運転するという方策を学習しているので、道からそれないようにどんどん走っていきます。よって軌道計画というのは特になくても、道の状況と実際の車の現在の状況を入れてあげることで動かすことができます。そういうことから、強化学習は軌道計画や運動学習を全部同時に考えていく枠組にもなっているのではないかと最近では思っています。

2.3 強化学習の要素

強化学習の要素には、

１．Policy　２．Reward function　３．Value Function

の３つがあります。Policy というのは ”行動を決めるルール ”です。例えば Random Policy といったら、迷路の問題で上・下・右・左 4 つの方向に動ける状況において、上・下・右・左に行く確率が等確率で起こるというものです。この Policy を決めることで目的に達することができます。Reward function というのは ”ある状態において獲得できる報酬 ”のことをいいます。先程の車の例でいうと、道から外れたら報酬は－１、道の上をどんどん走れていったら＋１づつ、というようにルールを与える事です。仮に、できるだけ長く走ることを目的とすると、この報酬を最大化する

ことが長く走ることになります。どういう報酬を与えるかというのも自分で決めなければならないのです。Value Function というのは、”将来にわたって獲得できる報酬の総和 ”のことをさし、状態の価値（State-Value Function）を判断することと、行動の価値（Action — Value Function）を判断することの 2 種類があります。

これら 3 つの要素、 Policy、Reward function、Value Function を決めることで強化学習を行うことができるのです。

＜演習＞　演習問題の前に、n-Armed Bandit 問題について説明します。

・The n-Armed Bandit Problem　まず、スロットマシーンが n 個あって、そのレバーを引くと、いくつかはわからないですがコインがでてくる状況を考えます。この時、例えば 1000 回プレイしてコインをできるだけたくさん出すにはどういう方策をとったらいいのか、という問題が n-Armed Bandit 問題です。1 回 1 回レバーをひくことをプレイと呼びます。それぞれのプレイでレバーをひくとコインがでてくるので、そのコインが報酬として与えられます。これは例えば平均的には 5 個出るとしても、確率的なので分散によって 2 個だったり 6 個だったりいろいろでてきます。よっていくつでてくるかというのは、経験から予想しなくてはなりません。すべてのスロットマシーンがどれだけでるかということがわかれば（推定できれば）どれを選択すればいいのかということがおのずと決まってくるので、行動価値を推定するという問題になります。

このときに 2 つの言葉、Explore と Exploit というのがあります。平均的は 5 個でてくるとしても、4 個だったり 6 個だったり 7 個だったり 1 回に出る数はいろいろなので、実際にでてきた値だけを信じていると、レバーを 1 回引くだけでは 1 番最大の報酬が出てくるものを探すということはできません。ですから、色々な行動を探索します（Explore)。今までの経験では 2 番目のスロットマシーンがたくさんでてくるのだけれども、他にもっといいものがあるのではないかと思って隣のやつを引いてみる。人間でもいろいろな行動を探索しますが、そういったことです。だけど、いつも適当にやっていると最大の報酬を得られることはありえません。ですから最も良い行動を利用しなくてはならない (Exploit) のですが、数をこなさないと確率的にどれがどのくらいでてくるかはわからないので、実際にレバーを引いてみるといった行動（Explore）も必要です。

ですから、Explore、Exploit 両方必要なのですけれども、Exploration と Exploitation をどれだけの割合でしたらいいかといった問題がでてきます。Exploitation というのは一番いいものだけを選んでくることですが、たまにはそうでないのも選んでみたとき（Exploration）に報酬がどれだけ返ってくるかというのを評価することでよりよい評価関数（Value Function）を作ろうというような方策をとります。　ここで Q（行動価値）をどうやって推定するかという話になります。例えばさっきの例で t 回レバーを引いたとき、その中で 2 番目のスロット選択（行動 a）を k_a 回選択したとすると、

$$
\begin{aligned}
Q_t(a) &= \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a} \\
\lim_{ka \to \infty} Q_t(a) &= Q^*(a)
\end{aligned}
$$

となります。ここで、$r_1, r_2, \cdots, r_{k_a}$ は各回の報酬です。これはそれぞれでてきた報酬をたして Ka 回で平均を取ったものに対し、Ka を無限大にすることによりスロットマシーンの価値関数がわかるので、こういった方法で Action-Value の推定を行うというやり方があります。

行動の乱雑性導入の仕方にはいくつか方法がありますが、その中に ϵ — greedy という方法があります。これは $1 - \epsilon$ の確率で Exploit するのですが、ϵ の確率だけ Explore するというものです。ϵ — greedy というのは Exploration と Exploitation をバランスさせる一つの方法になります。

では ϵ をどうやって決めたらいいのか、どれだけランダムにやったらいいのかという話しがでてきます。それは環境によっても依存するのですけれども、ϵ を変えると得られる報酬がどう変わるかを演習を通じてみてもらいたいと思います。

＜演習課題＞

ϵ-greedy によって The n-Armed Bandit Problem 解く MATLAB プログラム narmband.m を用いて、各確率 ϵ を変化させた場合の学習曲線をプロットせよ。また、もし 1000 回プレイする場合に、最も多くのコインを得るには ϵ をどのように設定すればいいだろうか。

＜結果＞　（鮫島先生の結果の図をのせる）

このグラフは ϵ をパラメータとしたときの、プレイを 1000 回と限定してもらえる報酬を最大にしようと考えたときの行動価値 Q のグラフです。X 軸が ϵ、Y 軸が Q を表しており、Q は 100 回平均したものになっています。このグラフから、$\epsilon = 0.07$ 付近で最大になっていることがわかります。すなわち、$\epsilon = 0.07$ がこの場合最適な ϵ であると考えられます。

Q この場合、解析的にどの ϵ が最適なのか求められないのですか？

答 スロットの特徴や再帰的な報酬の値の分布がわかっていれば求められるかもしれません。

3　環境との相互作用

3.1　Return for Continuing Tasks

今の n-Armed Bandit 問題はすごく単純な例で、状態といいうものがなくいつも同じ状態にいたようなものです。例えば部屋がいくつかあってそれぞれスロットマシーンがおいてあり、どの部屋のスロットマシーンがいいのか選ぶようになっているというものです。ここでは環境との相互作用で学習していくという話について述べたいと思います。

Agent が何か行動を起こすと報酬が得られますが、それと共に状態も更新される、という状況を考えます。そうするとある状態 S_t である行動 a_t を起こすと、報酬 r_{t+1} がもらえて状態が S_t から S_{t+1} になり、そこでまた新しい行動 a_{t+1} とをとると報酬 r_{t+2} が得られ、状態も S_{t+2} に変わる、というように状態と行動が遷移します。　これから $\pi_t(s, a)$ という変数をもちいますが、これは状態 s で a をとるという確率を表します。強化学習は経験からいかに方策 (policy)π を改善するかという問題です。状態がずっと遷移していって、それぞれの状態遷移のときに、おのおの報酬がもらえていたわけですけれども、長い試行を通して一番報酬が多く得られるような方策を決めるというのが問題になります。　またタスクとして、コンティニュアスタスクとエピソードタスクという 2 つを考えないといけませんが、discount factor γ を入れることにより同じように考えることができます。つまり、報酬を

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdot \\ &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \end{aligned}$$

とすれば、γ を 0 から 1 の間にすることですごく遠い将来のものは、階乗できいてきますのでほぼゼロになって、R は無限大に発散することがなくなって和が求まることになります。この式は ∞ の部分を T(最終時刻）にすることで、γ を 1 にしてエピソードタスクを表現できることになります。しかし、先程の ϵ と同様に γ もどう決めるかというのは一般的には分からないので、試行錯誤的に決めているというのが現状です。

3.2 マルコフ性

強化学習というのはマルコフ性というのを仮定しています。本当は $Pr\{s_{t+1}=s', r_{t+1}=r|s_t, a_t, r_t, s_{t-1}, a_{t-1}, \cdots, r_1, s_0.a_0\}$ なのですが、マルコフ性を仮定して t+ 1 の応答は 1 個直前の状態とアクションによって決める、すなわち

$$Pr\{s_{t+1}=s', r_{t+1}=r|s_t, a_t\}$$

としています。この場合、状態とアクションのセットが与えられると、ある状態である行動をとったときに次の状態 s' に行く確率を考えることができます。それは遷移確率（$P^a_{ss'}$) と呼ぶものです。また、s という状態で a という報酬が得られて s' に行ったときに、得られる報酬の期待値というのを $R^a_{ss'}$ と書きます。遷移確率と報酬の期待値があると、先程の迷路の問題のように右・左・上・下に行くといった 4 つのパターンしか動けないものを考えた場合、現在のますより右に行くのを決定論的にしたければ、右に行く確率を 1 にすればいいし、確率的に動かしたければ 0.9 の確率では右に行きあとの 0.1 の確率はランダムに動いてどこに行くかはわからないという条件にすればいいといったことになります。そういう風に確率が分かっていると状態がどのように遷移していくのかが記述できるわけです。その時にどれだけ報酬がもらえるかというのがわかっていれば、先程 s_t で a_t をとったときに s_{t+1} に行くといった状態遷移がありましたが、これが全部記述できることになります。また $R^a_{ss'}$ がわかっていれば、環境に相互作用してぐるぐる回っていくときにどういう状態遷移をして、そのときにどういう報酬を得られるかというのが全てわかります。

3.3 Valuen Functions

$R^a_{ss'}$ がわかっていればどういう報酬を得られるかというのが全てわかりますが、そうすると価値関数が計算できます。価値関数には状態価値関数と行動価値関数の 2 つがあります。状態価値関数は

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t|s_t=s\} \\ &= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\} \end{aligned}$$

と書け、t の時刻におけるある状態 s について、それ以降どれだけ報酬がもらえるかという価値関数を表しています。ここで、時刻 t で状態 s にいることが条件となります。また、「状態 s で行動 a をとると、どのくらい報酬が得られるか」を示す期待収益は、

$$\begin{aligned} Q^\pi(s,a) &= E_\pi\{R_t|s_t=s, a_t=a\} \\ &= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s, a_t=a\} \end{aligned}$$

と書けます。先程の n-Armed Banit Problem の場合は行動価値関数を決めていましたが、状態が無かったので、$Q(a)$ と表せていました。$V^\pi(s)$ と $Q^\pi(s,a)$ との違いは、a がついているかついていないかで、$V^\pi(s)$ の場合は状態 s において，方策 π で次々に行動した場合，得られる報酬がいいか悪いか、$Q^\pi(s,a)$ の場合は、s という状態で a をとってその後 π という方策に従った場合にどれだけ価値が得られるか、というものになっています。a というのが方策 π に関係ないということを覚えておいてください。それで $V^\pi(s)$ と $Q^\pi(s,a)$ との違いがわかります。

3.4 Bellman 方程式

Bellman 方程式を

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t|s_t = s\} \\ &= E_{\pi}\{r_{t+1} + \gamma V(s_{t+1})|s_t = s\} \end{aligned}$$

で定義します。この式は、すべての行動に対して、次におこると期待されるすべての状態における価値とその報酬の和を、生起確率で重みづけしたものです。この Bellman 方程式は

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \\ &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots) \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned}$$

から求めることができます。Bellman 方程式は確率の期待値で書いてありますが、これをはずして違う形で書くと、$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} R^a_{ss'} + \gamma V^{\pi}(s')$ となります。この式において、$\pi(s,a)$ は s という状態において a という行動をとる確率、すなわち方策を示します。例えば Grid Space (上下左右にのみ動ける格子平面) におけるランダム policy を考えるなら、$\pi(s, 上) = 1/4$, $\pi(s, 下) = 1/4$,$\pi(s, 左) = 1/4$, $\pi(s, 右) = 1/4$ です。右にのみ行くポリシーを考えるなら、$\pi(s, 右) = 1$ となります。また、$P^a_{ss'}$ は s という状態で、行動 a をとった時 s' に行く確率です。最後に求まる $V^{\pi}(s)$ は、π という方策のもとで、状態 s がどのくらい価値があるのかを示します。同様に Q についても π という方策のもとで、価値を求めることができます。

3.5 最適価値関数

これまでは、π という方策の元で、V、Q がどうなるのかを考えてきました。しかし実際に求めたいのは、方策πが最適でない時、得られる報酬を最大にする方策 π です。そこで、その最適な方策 π を求める方法を考えます。有限 MDP において、

$$\pi \geq \pi' \text{ if and only if } V^{\pi} \geq V^{\pi'} \text{ for all } s \in S \tag{1}$$

が成り立ちます。つまり、もし π' より π をとると価値関数 V が上がるのであれば、そちらの方策の方が良い訳です。ここで、私達は V 及び Q を最大にするπを求めたい。そこで最適な方策を π^* で表しましょう。この最適方策 π^* に対応して、最適状態関数 V^* が次の様に求められます。

$$V^*(s) = \max_{\pi} V^{\pi}(s) \text{ for all s } \in S \tag{2}$$

同様に、最適行動関数 Q^* が次のように求められます。

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \text{ for all } s \in S \text{ and } a \in A(s) \tag{3}$$

ここで、最適行動関数 Q^* は、ある状態 s で行動 a をとり、その後は最適方策にしたがって行動した時の期待収益を表します。

3.6 Bellman Optimally Equation

以上より、V についての Bellman 最適方程式は

$$\begin{aligned} V^* &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\ &= \max_{a \in A(s)} E_{\pi^*} \{ r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a \} \\ &= \max_{a \in A(s)} E_{\pi^*} \sum_{s'} P^a_{ss'} \{ R^a_{ss'} + \gamma V^*(s') \} \end{aligned}$$

と表せます。ここで、V^* はある状態における、最適な状態価値関数です。また、Q についても同じことが言え、

$$\begin{aligned} Q^*(s,a) &= E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | s_t = s, a_t = a \right\} \\ &= \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma \max_{a'} Q^*(s', a') \right) \end{aligned}$$

として、最適行動価値関数 $Q^*(s,a)$ を求めることができます。

最適状態価値関数 $Q^*(s,a)$ は、状態 s のもとある行動 a をとった時どれほどの報酬が得られるかを示すものなので、Q^* が与えられれば、先読みする必要もなく最適な方策 π^* が与えられ、

$$\pi^*_s = \arg \max_{a \in A(s)} Q^*(s,a)$$

となります。

3.7 Bellman Optimally Equation を解く

これより、具体的に Bellman 最適方程式を解くことで、最適な方策πを求めることができます。このようにしてπをもとめる事ができるなら、これを「強化学習」などと呼ばなくても良いのではと思われるかも知れません。しかし、我々はこれまでに色々な仮定をしました。

1. 環境のダイナミクスの知識が必要
2. 十分な計算資源
3. マルコフ性

1、はある状態で行動 a をとったらどこへ行くかや、どのような報酬があたえられるかに関する知識のことを言っており、これが判れば方策πを計算することが可能です。しかし、現実の問題を考えると、これが判っていることはほとんどないのです。2、はアクションの数や、その後にとりうる状態の数が増えた時の計算資源のことを言っています。これらの量が増すと、いくら 3、でマルコフ性を仮定しても計算資源は足りなくなります。このような仮定のもと、具体的に Bellman 最適方程式を解くには次の二つの方法あります。一つは動的計画法、もう一つは強化学習です。

- 動的計画法
  Bellman 最適方程式を解く方法です。しかし、状態が増えると計算量が増え、現実的に解く事はできなくなります。

- 強化学習
  Bellman 最適方程式を近似的に解く方法です。ただし、必ず正しい解を出すと言う保証はなく、もし、めったに起きない状態の中に最適解があるとすると、それを捜し出すことはできません。

4 動的計画法

4.1 Policy Evaluation

これから、π^* を求める方法の一つである動的計画法で、Bellman 最適方程式をどのように解くかについて話していきます。まず、Policy を評価することを行います。つまり、Policy π が与えられた時、その状態価値関数 V^π を計算すると言うことです。これは、先程と同様に以下の式で表すことができます。

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t|s_t = s\} \\ &= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} \end{aligned}$$

これより、V^π についての Bellman 方程式は同様に
$V^\pi(s) = \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}R^a_{ss'} + \gamma V^\pi(s')$ と書けます。この式を解いて V^π を求めます。

4.2 Iterative Methods

上の式より、V を徐々に変えてゆき、V^* を求める事を考えます。$V_0 \to V_1 \to \cdots \to V_t \to Vt+1 \to \cdots \to V^*$ ここで、各々状態を更新するときは、すべての行動 a について sweep します。V は大域的な計算式になっているので、下式にしたがって変わっていきます。
$V_{t+1} = \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}R^a_{ss'} + \gamma V_t(s')$ すなわち一つ前の V を使って、次の時刻の V を計算することができます。この計算を繰り返すことで、V^* をもと求めることができるのです。

4.3 Policy Improvement

次は、Q について考えます。すなわち、方策 π のもとで計算された V^π を用いて、状態 s で行動 a を行う事の価値を計算します。ここで、Q^π は

$$\begin{aligned} Q^\pi &= E_\pi\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\} \\ &= E_\pi \sum_{s'} P^a_{ss'}\{R^a_{ss'} + \gamma V^\pi(s')\} \end{aligned}$$

の様に書くことができるので、V を使って Q を計算することができます。
V^π に関して greedy に行うとは、すべての Q について最大の価値を与える 方策を $\pi'(s)$ とする時、$V^{\pi'}$ が V^π より大きければ更新を行うということです。すなわち、下式で表される操作を行うことに相当します。

$$\pi'(s) = \arg\max_{\pi} Q^{\pi}(s,a)$$
$$= \arg\max_{\pi} \sum_{s'} P^{a}_{ss'} \{R^{a}_{ss'} + \gamma V^{*}(s')\} \quad \text{Then} \quad V^{\pi'} \geq V^{\pi'}$$

これを繰り返してゆき、最後に $V^{\pi'}$ が V^{π} と等しくなったら、そこで更新を止めます。この時、Bellman 最適方程式は

$V^{\pi}(s) = \max_a \sum_{s'} P^{a}_{ss'} R^{a}_{ss'} + \gamma V_t(s')$ for all $s \in S$ となり、そして $\pi' = \pi^*$ が最適方策となります。

4.4 A Small Grid World

例として、簡単な計算をやってみましょう。ここでは Matlab は使わず、手で解いてみます。

＜４×４のグリッドの図＞

まず、図のような４×４のグリッドを考えます。ここで、とり得る行動は

Action = up, down, right, left

の４つののみとし、さらに次の様な条件をおきます。

1. $\gamma = 1$ とする。
2. グリッド No. 1-14 を非終端状態とする。
3. 1, 15 を終端状態とする。
4. 行動 a は状態を変えない。
5. 報酬は終端状態にたどり着くまで-1 ずつ与えられる。

さらに、状態 s において行動 a が選択されたとき、状態 s' になる確率 $P^{a}_{ss'}$ は決定論的であるとし、これを境界処理に用います。例えば

$P^{right}_{5,6} = 1, P^{right}_{5,10} = 0, P^{right}_{7,7} = 1$ の様になります。状態 5 から右という方向を選択すると、6(右) へ状態が変化する確率は１ですが、それ以外の方向へは動けません。逆に右側が存在しない状態 7 から右という行動を選択すると、同じ状態 7 へ行ってしまいます。また、ポリシーは上下左右各々1/4 で、固定することにします。

すべてのグリッドについて計算する時間はありませんので、グリッド 1 について V を計算してみましょう。まず、初期状態評価関数は何でも良いですのですべてのグリッドについて 0 とします (図 1 左)。

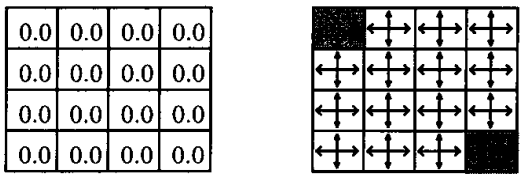


図 1: **初期状態**

このとき、グリッド 1 の状態は以下の式で更新されます。

$$\begin{aligned} V(1) &= \frac{1}{4}(-1+\gamma\cdot 0)+\frac{1}{4}(-1+\gamma\cdot 0)+\frac{1}{4}(-1+\gamma\cdot 0)+\frac{1}{4}(-1+\gamma\cdot 0) \\ &= -1.0 \end{aligned}$$

同様に他のグリッドも計算すると、(図 5 左) のようになります。

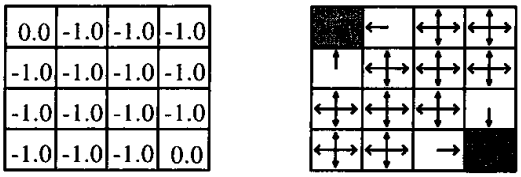


図 2: **1 回の計算後**

同様の計算を繰り返すと、グリッド 1 の状態は以下の式で更新され、全体の状態は (図 3 左) のようになります。

$$\begin{aligned} V(1) &= \frac{1}{4}(-1+\gamma\cdot 0)+\frac{1}{4}(-1+\gamma\cdot(-1))+\frac{1}{4}(-1+\gamma\cdot(-1))+\frac{1}{4}(-1+\gamma\cdot(-1)) \\ &= -1.75 \end{aligned}$$

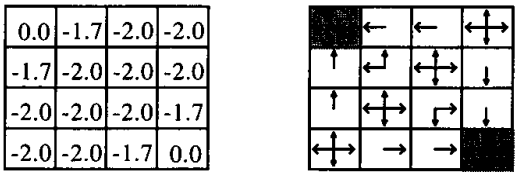


図 3: **2 回の計算後**

この計算を何度も繰り返すと、(図 4 左) の状態になります。この状態になると、式はもうこれ以上変化しません。

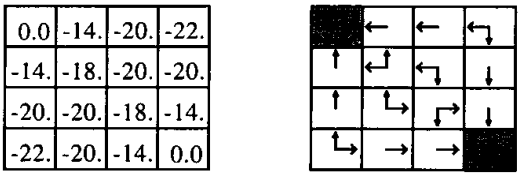


図 4: **最終状態**

求めた各状態 V を用いて一歩先読みし、もらえる報酬を予測することで、各状態からの最適行動を求めることができます (図あ、い、う、え 右)。例えば (図あ 左) の時はすべての状態価値 V が 0 ですので、ランダムに行動するしかありませんが、(図い)、(図う) になるにつれ、一歩先をみ

ることでどちらに行くのが良いか調べる事ができます。そして (図え) では V^* がわかっており、確実に最適な行動を見つける事ができます。しかし、この方法では、状態、行動の数が増えた時、計算が困難になります。

Q ゴールは無いのですか？

答 グリッド１と１５がゴールです。ここに行ったらおしまいです。

Q ゴールで報酬はないのですか？

答 無いです。代わりに、動くと-1 の報酬が与えられます。なるべく早くゴールした方が報酬の量が多い訳です。

Q 右に行けない場合も reward は-1 になるのですか？

答 そうです。

4.5 Generalized Policy Iteration

Small Grid World の場合は、一度ある方策 π^0 をおくだけで一番最適な $*$ が求まってしまいました。しかし、一般には方策の見積りと状態の見積りとを繰り返すことで V^*、π^* を求めます。すなわち

$\pi_0 \to V^{\pi_0} \to \pi_1 \to V^{\pi_1} \to \cdots \to \pi_* \to V^{\pi_*} \to$ として V^*、π^* を求めるのです (図、講義で用いた図 (下図))。

＜講義で用いた図＞

4.6 Monte Carlo 法

最適な方策 π^0 を求めるために、動的計画法の他にモンテカルロ法と呼ばれるものがあります。動的計画法 (DP) では、実際にはあり得ないすべての状態を考えますが、モンテカルロ法では一つの状態遷移のみを考えます。そして、ゴールにたどりつくまで一度試行を行ってみて、その時の報酬をもとに状態価値関数 $V(s)$ を更新するのです。迷路の例ですと、一度 agent がランダムに動いてでゴールにたどり着くまで試行を行い、それで得られる報酬に基づいて状態を更新することになります。

5 強化学習

5.1 TD Prediction

では、最後に強化学習における Bellman 最適方程式の解法について考えてみます。まず、方策 π が与えられた時、状態価値関数を計算する方法についてです。モンテカルロ法では下式の様に Terminal の状態まで試行を行い、時刻 t の後、実際に得られる報酬 R_t を用いて V を更新しまが、この方法ではゴールにたどり着くまで報酬が判らないので状態の更新ができません。

$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)]$ それに対して、強化学習では、何か行動があった段階で、V を更新する事を考えます。次の式は単純な TD(0) と呼ばれる強化学習のアルゴリズムです。$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$ ここで重要なのは、V の更新は収益の期待値 $r_{t+1} + \gamma V(s_{t+1})$ を用いて行われるという事です。これは、得られる報酬を予測したもので、MC で用いられている報酬 R_t と同種のものです。V は予測値なので、常に正しいとは限りません。時刻が一つずれてい

るので Temporal Difference (TD) と名付けられています。

Q R と V の違いは何ですか？

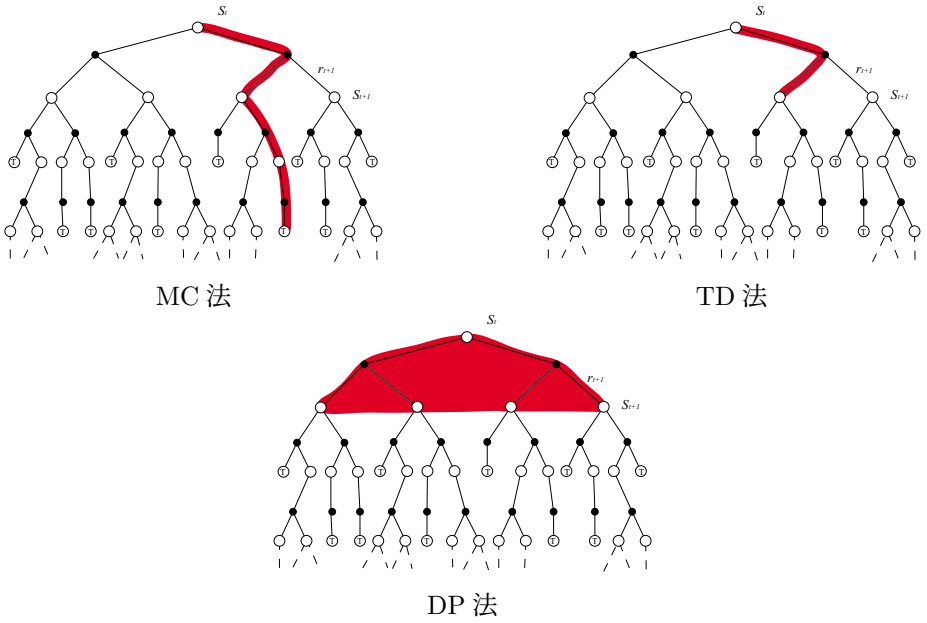
答 R は最終状態にいった時もらえるもの、V_{t+1} はその時の状態価値関数を用いて予測したものです。

Q 予測が完全ならば MC 法の R_t と TD 法の $r_{t+1} + \gamma V(s_{t+1})$ は一致するのですか？

答 はい。

5.2 Backup diagram

ここで、(図、下の図) にこれまで紹介した 3 つの方法の Backup diagram を示します。



MC 法　　TD 法

DP 法

図 5: **3 つの方法の比較**

「(MC)」では、ある状態から「Action →状態→ Action →状態→」を繰り返してゆき最後に Terminal で得られた報酬 R を用いて V を更新します。「(TD)」では、ある状態から「Action → 次の状態」という情報のみを用いて状態 V を更新します。「動的計画法 (DP)」では、ある状態で取りうる Action をすべて計算し、その結果得られる状態の価値関数をすべて足しています。つまり、すべての可能性を考えて更新していきます。(図、上の図) の赤い部分は、考慮にいれる状態、及び Action です。TD 法が一番小さい、すなわち状態の更新の為に考慮にいれる情報が少なくて良い事が判ります。

5.3 Q-learning

Q-learning とは、各状態において、可能な行動の中で、最も行動評価関数の値が高い行動をとるように学習を行う方法です。学習は次の様に行われます。$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$ Q-learning が方策 off 型と呼ばれるのは、方策に関係なく a について最大の行動価値関数 Q を用いてもとの Q と比較を行い、行動価値関数を更新するためです。

5.4 Actor-Critic

＜教科書 P.86 の図＞

Q-learning は、ある状態である行動をとることに対する善し悪しのみで方策 Q が決まりました。それに対して Actor-Critic は価値関数と同様に、学習する方策が陽に表現されます。また、行動選択に必要な計算量が少ない、Soft Max を使うなど確率的な方策も学習できる、そして生物学的理解に根ざしたモデルであるなどの特徴があります。Actor-Critic は、教師信号でなく、誤差信号を得ることで学習を行います。すなわち $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ で誤差信号 δ_t を求め、$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta_t$ で $p(s,t)$ (actor が状態 s で行動 a を取る確率) を更新します。δ_t は V が完全に報酬を予測する事ができるようになれば 0 になります。

Q 初日に銅谷さんが見せたシュルツの実験では、ドーパミンニューロンの活動がずれていました。これは TD 誤差なのですか？もし TD 誤差なら学習が進むとゼロになるはずですし、一方それが価値関数であるならずっと出ている筈ですが？

答 (銅谷) TD 誤差と考えています。学習が進んでいない段階では $V(s) = 0$ なので、$\delta_t = r_t$ が出ているわけです。

Q すると、これは学習が進行していないということですか？学習が進行すれば活動がゼロになるはずですね。

答 (銅谷) タスクが始まるまでは、サルはいつ reward が貰えるのか判りません。そのため、タスクが始まるところでは V が急に大きくなるのです。そして、1-2 秒後に小さくなりますが、これは reward があると予想できる様になる為です。

Q では、サルが自分で start ボタンを押して、課題をはじめる様な実験を行った場合はどうなるのですか？

答 (銅谷) 課題の始まりをサル自身に行わせるという事ですね。彦坂先生はご存知ですか？

答 (彦坂) 例えば Block 単位で trial を行う場合は、最初の trial の Fixation Point にサルは一番反応します。そして 2 trial 目以降はほとんど変わりません。

Q それは Inter Trial Interval に依存しませんか？

答 (彦坂) それは依ります。

Q 私はサルの訓練後、報酬が貰える筈のところで報酬が出なかった時の事を知りたいです。これも TD 誤差 (マイナスの誤差) になりますが、反応はでるのですか？もしくはマイナスの誤差ですから違うところで出るのですか？

答 (銅谷) それに関してはシュルツ達が実験していて、本来出るべきタイミングで reward が出ないと、一時的にドーパミンニューロンに Depression が出ることが観測されています。