その他の主要技術(2.. 強化学習)

前田利之（まえだ　としゆき）
aipr@e-chan.jp

阪南大学　経営情報学部

(2024.1.12)

その他の主要技術(2.. 強化学習)

- 時間があれば取り上げたかった話題は沢山あるのですが、その中のうち重要と思われるもの（その２）として

  強化学習

  を、取り上げます
- 強化学習自体は深層学習と別の概念です（後述）

そもそも論：機械学習の分類

- 教師あり学習
  - 入出力の [　　] を学習する（[　　] データが与えられる）
  - 一般的にイメージする機械学習（特に Deep Learning）はこれが多い
- 教師なし学習
  - 正解データが与え [　　]
  - クラスタリング、異常検知などに応用

そもそも論：機械学習の分類

- 教師あり学習
  - 入出力の 関係 を学習する（　　データが与えられる）
  - 一般的にイメージする機械学習（特に Deep Learning）はこれが多い
- 教師なし学習
  - 正解データが与え　　
  - クラスタリング、異常検知などに応用

そもそも論：機械学習の分類

- 教師あり学習
  - 入出力の **関係** を学習する（ **正解** データが与えられる）
  - 一般的にイメージする機械学習（特に Deep Learning）はこれが多い
- 教師なし学習
  - 正解データが与え
  - クラスタリング、異常検知などに応用

そもそも論：機械学習の分類

- 教師あり学習
  - 入出力の **関係** を学習する（ **正解** データが与えられる）
  - 一般的にイメージする機械学習（特に Deep Learning）はこれが多い
- 教師なし学習
  - 正解データが与え **られない (!)**
  - クラスタリング、異常検知などに応用

機械学習の分類 (cont.)

- 強化学習←これが今日のテーマ
  - [　　]を通じて[　　]を最大化する
  - 入力は[　　]、正解データは[　　]
    （環境と報酬が与えられる…後述）
  - 人間が成長するようなもの（？）

機械学習の分類 (cont.)

- 強化学習←これが今日のテーマ
  - 試行錯誤を通じて　　を最大化する
  - 入力は　　、正解データは　　
    （環境と報酬が与えられる…後述）
  - 人間が成長するようなもの（？）

機械学習の分類 (cont.)

- 強化学習←これが今日のテーマ
  - 試行錯誤を通じて 成果（価値） を最大化する
  - 入力は [　　]、正解データは [　　]
    （環境と報酬が与えられる…後述）
  - 人間が成長するようなもの（？）

機械学習の分類 (cont.)

- 強化学習←これが今日のテーマ
  - 試行錯誤を通じて 成果（価値） を最大化する
  - 入力は 乱数、正解データは
    （環境と報酬が与えられる…後述）
  - 人間が成長するようなもの（？）

機械学習の分類 (cont.)

- 強化学習←これが今日のテーマ
  - 試行錯誤を通じて成果（価値）を最大化する
  - 入力は乱数、正解データは与えられない
    （環境と報酬が与えられる…後述）
  - 人間が成長するようなもの（？）

強化学習とは

- コンピューターエージェントが、動的環境と繰り返し [ ] のやりとりを重ねることによってタスクを解決できるようになる手法
  - エージェント: [ ] ←行動の主体
  - 環境: [ ]
  - タスク: 解決すべき [ ]
  - 行動: タスクを解決するため起こす (action)
  - 報酬: 行動をおこしたときの成果（指針）

強化学習とは

- コンピューターエージェントが、動的環境と繰り返し **試行錯誤** のやりとりを重ねることによってタスクを解決できるようになる手法
  - エージェント:  ←行動の主体
  - 環境: 
  - タスク: 解決すべき 
  - 行動: タスクを解決するため起こす (action)
  - 報酬: 行動をおこしたときの成果（指針）

強化学習とは

- コンピューターエージェントが、動的環境と繰り返し **試行錯誤** のやりとりを重ねることによってタスクを解決できるようになる手法
  - エージェント: **開発対象（モデル）** ←行動の主体
  - 環境:
  - タスク: 解決すべき
  - 行動: タスクを解決するため起こす (action)
  - 報酬: 行動をおこしたときの成果（指針）

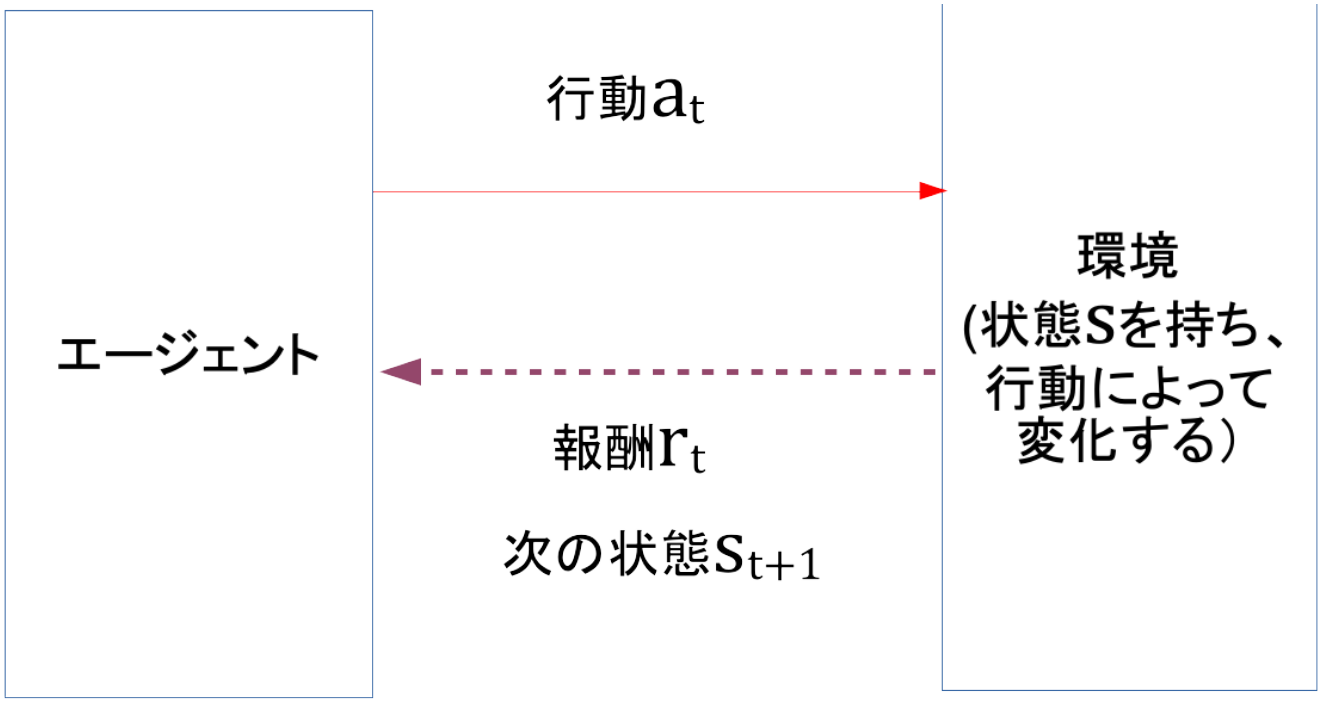
強化学習とは

- コンピューターエージェントが、動的環境と繰り返し **試行錯誤** のやりとりを重ねることによってタスクを解決できるようになる手法
  - エージェント: **開発対象（モデル）** ←行動の主体
  - 環境: **「状態」を持ち提供する**
  - タスク: 解決すべき
  - 行動: タスクを解決するため起こす (action)
  - 報酬: 行動をおこしたときの成果（指針）

強化学習とは

- コンピューターエージェントが、動的環境と繰り返し **試行錯誤** のやりとりを重ねることによってタスクを解決できるようになる手法
  - エージェント: **開発対象（モデル）** ←行動の主体
  - 環境: **「状態」を持ち提供する**
  - タスク: 解決すべき **課題**
  - 行動: タスクを解決するため起こす (action)
  - 報酬: 行動をおこしたときの成果（指針）

強化学習とは(cont.)



状態s_tを見て
行動a_tを決める

状態s_tと行動a_tから
報酬r_tと次の状態s_{t+1}を計算する

強化学習とは(cont.)

- この学習手法により、エージェントは、タスクの [ ] する一連の意思決定を行うことができる
- 原則的に、人間が介入したりタスクを達成するために明示的にプログラムしたりする [ ]
- [ ] のプログラムが強くなったのはこの技術のおかげ
  - 代表例: AlphaGo, AlphaZero (by DeepMind/Google)

強化学習とは(cont.)

- この学習手法により、エージェントは、タスクの**報酬を最大化**する一連の意思決定を行うことができる
- 原則的に、人間が介入したりタスクを達成するために明示的にプログラムしたりする
- のプログラムが強くなったのはこの技術のおかげ
  - 代表例: AlphaGo, AlphaZero (by DeepMind/Google)

強化学習とは(cont.)

- この学習手法により、エージェントは、タスクの**報酬を最大化**する一連の意思決定を行うことができる
- 原則的に、人間が介入したりタスクを達成するために明示的にプログラムしたりする**必要はない**
- [　　]のプログラムが強くなったのはこの技術のおかげ
  - 代表例: AlphaGo, AlphaZero (by DeepMind/Google)

強化学習とは(cont.)

- この学習手法により、エージェントは、タスクの**報酬を最大化**する一連の意思決定を行うことができる
- 原則的に、人間が介入したりタスクを達成するために明示的にプログラムしたりする
  必要はない
- **将棋、囲碁**のプログラムが強くなったのはこの技術のおかげ
  - 代表例: AlphaGo, AlphaZero (by DeepMind/Google)

強化学習の特徴

- データが必要無い←データは強化学習環境上で動くエージェント自身が [　　] して作成する
- データ至上主義の深層学習界では異色の存在
  - データでは GAFAM に勝てないけど、強化学習なら勝負できる（?!）
  - データが無くてもいろいろ試せるので楽しい（！）

強化学習の特徴

- データが必要無い←データは強化学習環境上で動くエージェント自身が 探索 して作成する
- データ至上主義の深層学習界では異色の存在
  - データでは GAFAM に勝てないけど、強化学習なら勝負できる（?!）
  - データが無くてもいろいろ試せるので楽しい（！）

Q学習 (Q-Learning)

- Q学習では、 [　　] を更新して学習を進める
- [　　] は、状態sで行動aを行ったときの収益を推定する関数
- イメージは、複数の状態 s_i と 行動 a_i の [　　] の構築

Q学習 (Q-Learning)

- Q学習では、 Q関数 を更新して学習を進める
- [　　] は、状態sで行動aを行ったときの収益を推定する関数
- イメージは、複数の状態 s_i と 行動 a_i の [　　] の構築

Q学習 (Q-Learning)

- Q学習では、 Q関数を更新して学習を進める
- Q関数は、状態sで行動aを行ったときの収益を推定する関数
- イメージは、複数の状態 s_i と 行動 a_i の　　の構築

Q学習 (Q-Learning)

- Q学習では、 Q関数を更新して学習を進める
- Q関数は、状態sで行動aを行ったときの収益を推定する関数
- イメージは、複数の状態 s_i と 行動 a_i の対応表の構築

Q学習 (cont.)

1. Q(s, a) を任意に初期化
2. 各エピソードに対し以下を繰り返す
   1. s を初期化
   2. エピソードの各ステップに対して繰り返す
      1. Q 関数と探索を使って、s での行動 a を選択する
      2. 行動 a を取り、r, s' を観測する
      3. Q(s, a) ← Q(s, a) + [r + Q(s', a') - Q(s, a)]
      4. s ← s'
   3. s が終端状態ならば繰り返しを終了

☆エピソード：強化学習に与えられた環境に対する行動（一連の試行）の開始から終了までの期間

多腕バンディット問題 (multi-armed bandit)

☆ https://qiita.com/ikeyasu/items/67dcddce088849078b85 参照

- 強化学習の典型的例題
- バンディット = [ ]
- マシンが複数あるとき、どのスロットをまわせば収益が最大化できるかを推定する問題
- 課題＝確率的に報酬を出すので、短期的に[ ]が上がったからと言って、そのマシーンばっかりまわすと[ ]につながらないかもしれない（ある程度の数を試さないといけない）

多腕バンディット問題 (multi-armed bandit)

☆ https://qiita.com/ikeyasu/items/67dcddce088849078b85 参照

- 強化学習の典型的例題
- バンディット = [ ]
- マシンが複数あるとき、どのスロットをまわせば収益が最大化できるかを推定する問題
- 課題＝確率的に報酬を出すので、短期的に[ ]が上がったからと言って、そのマシーンばっかりまわすと[ ]につながらないかもしれない（ある程度の数を試さないといけない）

多腕バンディット問題 (multi-armed bandit)

☆ https://qiita.com/ikeyasu/items/67dcddce088849078b85 参照

- 強化学習の典型的例題
- バンディット = スロットマシン
- マシンが複数あるとき、どのスロットをまわせば収益が最大化できるかを推定する問題
- 課題＝確率的に報酬を出すので、短期的に［　　］が上がったからと言って、そのマシーンばっかりまわすと［　　］につながらないかもしれない（ある程度の数を試さないといけない）

多腕バンディット問題 (multi-armed bandit)

☆ https://qiita.com/ikeyasu/items/67dcddce088849078b85 参照

- 強化学習の典型的例題
- バンディット = スロットマシン
- マシンが複数あるとき、どのスロットをまわせば収益が最大化できるかを推定する問題
- 課題＝確率的に報酬を出すので、短期的に報酬が上がったからと言って、そのマシーンばっかりまわすと　　につながらないかもしれない（ある程度の数を試さないといけない）

多腕バンディット問題 (multi-armed bandit)

☆ https://qiita.com/ikeyasu/items/67dcddce088849078b85 参照

- 強化学習の典型的例題
- バンディット = スロットマシン
- マシンが複数あるとき、どのスロットをまわせば収益が最大化できるかを推定する問題
- 課題＝確率的に報酬を出すので、短期的に報酬が上がったからと言って、そのマシーンばっかりまわすと収益の最大化につながらないかもしれない（ある程度の数を試さないといけない）

多腕バンディット問題 (cont.)

- エージェント：
- 環境：
- 行動 a_i：i 番目のスロットマシンを
- 報酬 r：スロットマシンの

多腕バンディット問題 (cont.)

- エージェント：プレイヤー（人間）
- 環境：
- 行動 a_i：i 番目のスロットマシンを
- 報酬 r：スロットマシンの

多腕バンディット問題 (cont.)

- エージェント：プレイヤー（人間）
- 環境：スロットマシン
- 行動 a_i：i 番目のスロットマシンを
- 報酬 r：スロットマシンの

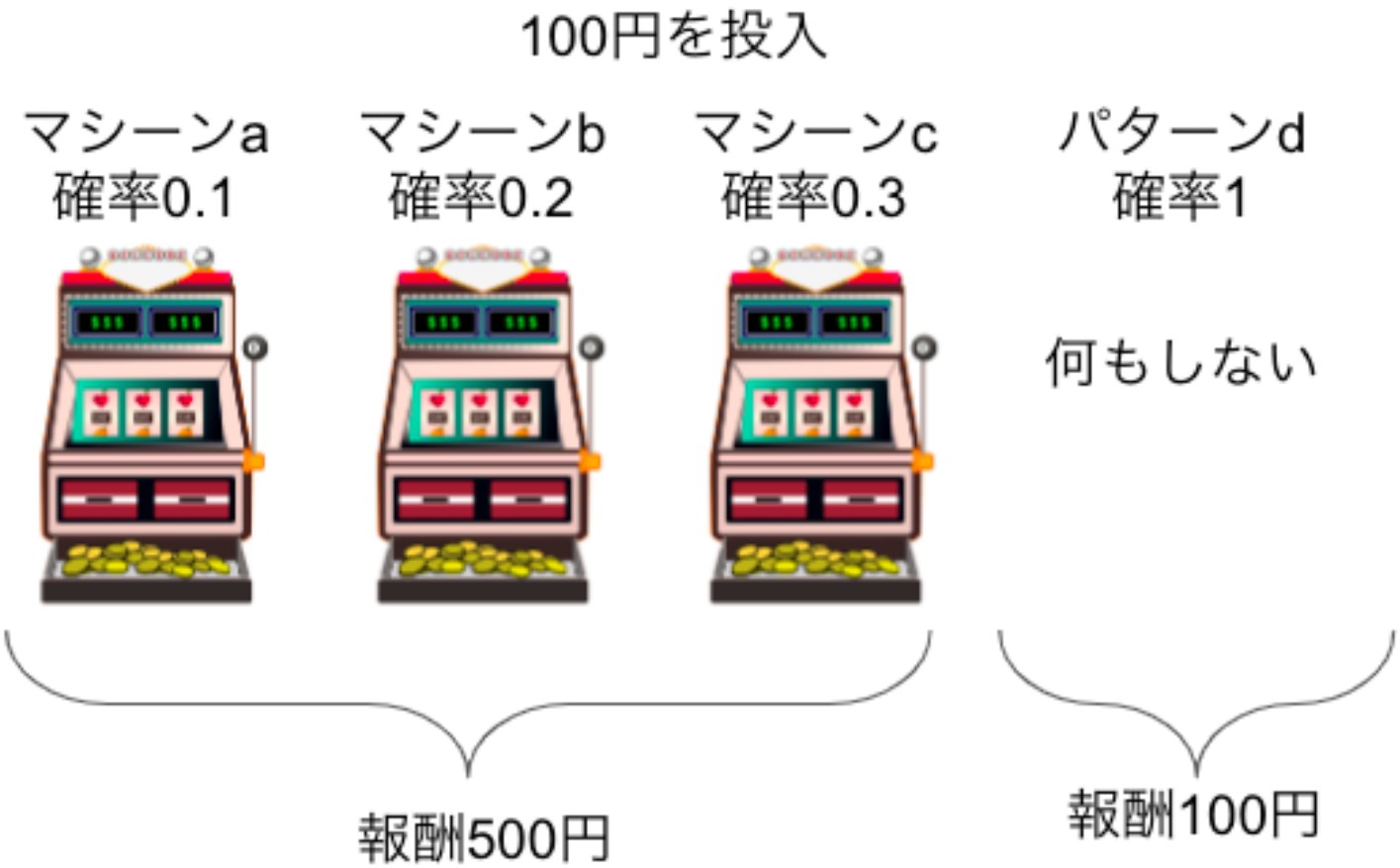
多腕バンディット問題 (cont.)

- エージェント：プレイヤー（人間）
- 環境：スロットマシン
- 行動 a_i：i 番目のスロットマシンを選んでレバーを引く
- 報酬 r：スロットマシンの

多腕バンディット問題 (cont.)

- エージェント：プレイヤー（人間）
- 環境：スロットマシン
- 行動 a_i：i 番目のスロットマシンを選んでレバーを引く
- 報酬 r：スロットマシンの払い戻し額

多腕バンディット問題 (cont.)



多腕バンディット問題 (cont.)

- ベットは１回100円
- （…ちょっと単純化）
- ２台のスロットマシンがあると仮定
  - １台目：確率 0.1 で報酬 500 円（期待値 50 円）
  - ２台目：確率 0.4 で報酬 500 円（期待値 200 円）
  - (3 つめの選択＝賭けない：確率１で報酬＝期待値 100 円)

… □台目を選ぶのが正解となる、はず

多腕バンディット問題 (cont.)

- ベットは１回100円
- （…ちょっと単純化）
- ２台のスロットマシンがあると仮定
  - １台目：確率 0.1 で報酬 500 円（期待値 50 円）
  - ２台目：確率 0.4 で報酬 500 円（期待値 200 円）
  - (3 つめの選択＝賭けない：確率１で報酬＝期待値 100 円)

… 2台目を選ぶのが正解となる、はず

多腕バンディット問題 (cont.)

- たとえば、最初３回をランダムに引いて、報酬の平均が最大のマシン（あるいは何もしない）を選択することにすると、最初のランダムで１台目あるいは「何もしない」が選択されると、そのままになる（実装例 1-1 参照）
- → Epsilon-Greedy 法（実装例 1-1 参照）
  - 確率 ϵ で探索、つまりランダムに行動 a を選び、確率 (1-ϵ) で従来通り Q 値が最大のものを選ぶ
  - この ϵ は一般的にステップ毎に少しずつ減らす（例えば、最初は 1(完全にランダム) で、最終的に 0.1 程度にする、等）

多腕バンディット問題（実装例）

1. まず、bandit-2.ipynb (zip で圧縮済) のリンク先を Colab Notebooks の中（ mymodules の中ではない！）に保存し、展開しておく
2. Chrome または Edge を立ち上げ Google Drive からファイルをひらく
3. ２分割されている。１つ目は前頁 1-1 である。何回か実行して、 a,b,c の何れかが選択されることを確認する
4. ２つ目は 1-2 の実装である。結果として b が選択されることを確認する

強化学習のメリット

- 学習の過程で行動の評価方法自体を学習するため、一度の行動に対する評価尺度の定義が [　　] 問題についても扱うことができる
- ⇒ [　　] の環境に対する適応が期待できる

強化学習のメリット

- 学習の過程で行動の評価方法自体を学習するため、一度の行動に対する評価尺度の定義が 難しい 問題についても扱うことができる
- ⇒ [　　] の環境に対する適応が期待できる

強化学習のメリット

- 学習の過程で行動の評価方法自体を学習するため、一度の行動に対する評価尺度の定義が **難しい** 問題についても扱うことができる
- ⇒ **未知** の環境に対する適応が期待できる

強化学習のデメリット

- どのような評価方法で、どのような行動を学習するかは実際の学習モデル［　　］になる
- その結果、人間が［　　］ような行動をとったり、性能改善の為にどのように手を加えるべきかを調査するのが［　　］になる可能性がある

強化学習のデメリット

- どのような評価方法で、どのような行動を学習するかは実際の学習モデル 任せ になる
- その結果、人間が [　　] ような行動をとったり、性能改善の為にどのように手を加えるべきかを調査するのが [　　] になる可能性がある

強化学習のデメリット

- どのような評価方法で、どのような行動を学習するかは実際の学習モデル **任せ** になる
- その結果、人間が **意図しない** ような行動をとったり、性能改善の為にどのように手を加えるべきかを調査するのが [　　] になる可能性がある

強化学習のデメリット

- どのような評価方法で、どのような行動を学習するかは実際の学習モデル**任せ**になる
- その結果、人間が**意図しない**ような行動をとったり、性能改善の為にどのように手を加えるべきかを調査するのが**困難**になる可能性がある

深層Q学習 (Deep Q-Learning, DQN)

- 複数の状態 s_i と 行動 a_i の対応表は、状態と行動が多数になると複雑になって［　　］になる
- →［　　］で学習させる
- ゲーム AI は、ほとんどがこれ
- （以下、詳細略）

深層Q学習 (Deep Q-Learning, DQN)

- 複数の状態 s_i と 行動 a_i の対応表は、状態と行動が多数になると複雑になって計算が大変になる
- → [　　] で学習させる
- ゲーム AI は、ほとんどがこれ
- （以下、詳細略）

深層Q学習 (Deep Q-Learning, DQN)

- 複数の状態 s_i と 行動 a_i の対応表は、状態と行動が多数になると複雑になって計算が大変になる
- → ニューラルネットワークで学習させる
- ゲーム AI は、ほとんどがこれ
- （以下、詳細略）

DQN （実装例…ブロック崩し）

☆ https://github.com/toshikwa/rl-tutorials/blob/main/dqn.ipynb 参照

1. まず、dqn-ex.ipynb (zip で圧縮済) のリンク先を Colab Notebooks の中（ mymodules の中ではない！）に保存し、展開しておく
2. Chrome または Edge を立ち上げ Google Drive からファイルをひらく
3. いくつかに分割されているので、上から順番に実行し、結果を確認する
4. 【注意】場合によっては１時間以上かかるので、授業中に終わらない場合は別途（時間のあるときに）やりなおしてください（！）

おしまい

質問・コメント等あれば、何でもお気軽に
aipr@e-chan.jp に
メールください！