2020・09・11　ゼミ資料

Pythonで学ぶ強化学習

2章 強化学習の解法(1):
環境から計画を立てる

1116 17 9036

山口真哉

やること

- chapter 2 ではchapter 1 で実装した迷路の環境をベースに計画を立てる手法を学ぶ.
- 計画を立てるためには「価値評価」と「戦略」が必要 ⇒「価値」を実態に即した形で定義しなおす.
- そのため,「価値の定義」,「価値評価の学習」,「戦略の学習」の3ステップで進める.

chap 2

- chapter 2 で扱う学習手法は動的計画法(DP)である.
  - 計算量を状態数(に線形)で抑える
- 動的計画法の例としてナップサック問題やLCS, ゲームDPなど色々ある.
- DPは遷移関数と報酬関数が明らかな場合に使える.
- このような学習手法をモデルベースといい, 対義語としてモデルフリーがある.
- モデルフリーは例えば将棋とかがそれにあたる. (chapter 3以降で触れる)

ベルマン方程式

- 価値は以下のような式で表された.

$$G_t := \sum_{k=t+1}^{T} \gamma^{t+1-k} r_k = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

- この価値には2つの問題点がある.
  - 将来の即時報酬の値が判明している必要がある点
  - それらが必ず得られるとしている点

- 例えば各時間にサイコロを1回振って出た目の大きさ分だけ報酬が得られるとして,
  サイコロを振ってみるまで得られる報酬はわからない上に得られる報酬は確率的.
  (「各時間にどの目が出るか予想ができていて, それが必ず当たる.」というわけではない.

ベルマン方程式

- 1点目は G_t が再帰的な構造を持っていることに注意して G_{t+1} の計算を持ち越すことや, G_{t+1} に適当な値を入れて G_t を計算することが可能になる.

$$G_t = r_{t+1} + \gamma G_{t+1}$$

- 2つ目の問題点は即時報酬に行動確率をかける(期待値を算出)することで解決できる.

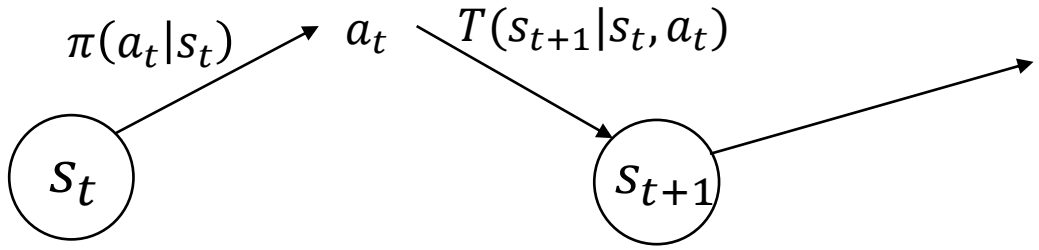
- ここで行動を定義する方法は2つある.
  - エージェントは保持している戦略に基づき行動する.
  - エージェントは常に「価値」が最大になる行動を選択する.

- 以上を踏まえて, 状態 s から戦略 π に基づいて行動する価値 $V_\pi(s)$は以下で定義できる.

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

ベルマン方程式

戦略 π に基づいて行動する場合, 状態 s で行動 a をとる確率は $\pi(a|s)$ となる.
そして, 遷移先 s' へは遷移確率から導かれる確率は $T(s'|s,a)$で遷移する.



報酬を $R(s,s')$ で書き直すと以下が成立.

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

期待値を行動, 状態に関して展開

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

「価値」を再帰的かつ期待値で表現することで2つの問題点をクリアした.
この式をベルマン方程式という.

ベルマン方程式

価値が最大になる行動を常に選択する場合も同様に,

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

特に報酬が状態のみで決まるとき,

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

が成立する.

戦略に基づき行動するか, 価値が最大になるように行動するかは
強化学習の手法を分類する際に重要な観点になる.
前者を Policyベース , 後者を Valueベース と呼ぶ.
行動の評価方法のみを学習し評価単体で行動を決定するのがValueベースで,
Policyベースでは戦略で行動を決定し, その評価, 更新に行動評価が使われる.

実際の学習 (Valueベース)

- 「各状態の価値を算出し, 値が最も高い状態に遷移する行動をする」という考え方がValueベースの基本的な考え方
- DPによって各状態の価値を算出する方法を, 価値反復法 (Value Iteration)と呼ぶ.
- 価値反復法ではベルマン方程式の解を複数回繰り返して値の精度を高めていく.

$$V(s) \leftarrow \max_a \sum_{s'} T(s'|s,a)(R(s,s') + \gamma V(s'))$$

実際には $V(s)$ の更新差異が小さな値 ε 未満になるまで学習を続ける.

実際の学習 (Policyベース)

- 「エージェントは保持している戦略に基づき行動する」という考え方が
  Valueベースの基本的な考え方
- 戦略は状態での行動確率を出力するが, この行動確率から価値の計算が可能になる.
- 「戦略により価値を計算し, 価値を最大化するように戦略を更新する.」
  というプロセスを繰り返すことで, 価値の値, 戦略の双方の精度を上げていく.
- この繰り返しのプロセスを Policy Iteration と呼ぶ.

Valueベース? Policyベース?

- Policy IterationはValue Iteration に比べて全状態の価値を計算しなくてよい.
- しかしPolicy Iterationは戦略が更新されるたびに価値を計算しなければならない.
- そのためどちらが速く収束するかはケースバイケースとなる.

モデルベース, モデルフリー 1

- chapter 2の動的計画法を行う中で, エージェントが一切動いていなかった.
- 実際にコード中でも変数 agent_state を一切使わなかった.
- モデルベースではタイトルの「環境からのみ計画を立てる」が示すように,
  エージェントは一歩も動かず環境の情報から最適な戦略を得ている.
- このようなことが可能なのは遷移関数と報酬関数が明らかであるため. (実際に動かす必要がない)
- この手法はエージェントを動かすコストが高い場合や, 環境においてノイズが入りやすい場合に有効.
- ただし, 遷移関数・報酬関数の適切なモデル化が必要になる.

モデルベース, モデルフリー 2

- モデルフリーでは実際にエージェントを動かして経験から計画を立てていく.
- これは, 遷移関数・報酬関数が不明な場合でも適用できる.
- 一般的に遷移関数・報酬関数が既知であるケース, うまくモデル化できるケースが少ないため
  モデルフリーがよく使われている. (現在は深層学習の影響でモデルベースも増えてきた)
- またモデルベースとモデルフリーは対立する手法ではなく併用することも可能.

まとめ

- 「価値の定義」,「価値評価の学習」,「戦略の学習」を順に見てきた.
- 価値の定義ではchapter 1 の2つの問題点を再帰, 期待値を使って解消した.
- これを使ってベルマン方程式を導出した.
- 価値評価の学習(Valueベース)は価値の値のみで行動を決定した.
- 戦略の学習(Policyベース)は戦略をもとに行動を決定した.
- 動的計画法により, 状態数を減らして計算量を減らした.
- 実際にコードを動かして理解を深めてほしい.

https://github.com/masa-aa/Reinforcement-Learning-with-Python/tree/master/chap2

何か質問はありますか？