総合情報学実習 金子研究室最終レポート

08-182024 教養学部学際科学科B群総合情報学コース 3年 松井誠泰

実習の概要

今回は、研究室の体験として、様々なゲーム課題を機械学習、特に強化学習を用いて解決するというテーマのもと、深層強化学習によるOpenAIGymのゲーム課題解決を目的とした実習を行った。深層強化学習に必要となる基礎的な仕組みを理解できるよう、できるだけ機械学習用ライブラリを用いず全体の実装を行うこととした。

目的

今回の実習の目的を明確に書き下すと、以下のようになる。

- ライブラリを使用しない深層強化学習の実装
- 実装した深層強化学習ネットワークによるFrozenLake-v0, Taxi-v2課題の実行

方法

解析的な微分による誤差逆伝播の実装

まずは、ネットワークの単位となるパーセプトロンの順伝播、逆伝播を実装した。初めはパーセプトロン単位での実装を行ったが、後半ではネットワーク一層ごとに実装することで、重みをベクトルではなく行列で表現し、Pythonの繰り返し演算をNumpyの行列演算に置き換えることができ、高速化を実現し、学習を試す回数を増やすことができた。以下のような計算を行い、誤差（損失）関数を最小化するため、連鎖律を利用して前層までの勾配に現在の層の勾配をかけたものを現在の重みから引くことを連鎖的に繰り返す誤差逆伝播法を実装した。

O：学習対象の正しい出力ベクトルを集めた行列

X：学習対象の入力ベクトルを集めた行列

W：重みの行列

$$L(W, X) = (sigmoid(W \cdot X) - O)^2$$
$$\frac{\partial L(W, X)}{\partial W} = 2 \times X^T \times sigmoid(W \cdot X) \times (1 - sigmoid(W \cdot X)) \times (sigmoid(W \cdot X) - O)$$

実装したNetworkのテスト

ランダムに生成した入力に対し、XOR関数を用いた出力を求め、このデータを学習させることで、関数をシミュレーションし、深層学習ネットワークのテストを行った。XOR関数を用いた理由は、線形分離不可能なため、単純な識別関数やパーセプトロン単体では学習できない課題であり、ネットワークの表現能力をテストするのにちょうど良い課題だからである。また、適宜数値微分との誤差を計算し、勾配確認を行った。

深層強化学習の種々の手法の実装

正則化

初めは、実装の際、関数に重みを足したものを出力としていたが、実装のミスが目立ったため、最終的に使用した関数からは省略することとなった。考察でも挙げるが、適切な正則化を行わなかったことによる問題が多かった。

Experience replay

深層強化学習は、一般的な強化学習と異なり、学習のために表引きを行わず、代わりにニューラルネットワークを使用するため、通常の強化学習と同じ様に、行動した順に結果を学習していくと、時系列についても学習を行ってしまう。強化学習は、状態に対して、一意に行動価値が決定されることが望ましいので、Expericence replayと呼ばれる、学習データを一定量集めたのちに、それらからランダムに学習データを選び、バッチ学習を行う手法を用いる。

Target network

深層強化学習は、一般的な強化学習と異なり、学習のために表引きを行わず、代わりにニューラルネットワークを使用する。そのため、ある状態に対する行動価値の学習を行う際、他の状態や行動に対する価値の計算結果にも影響が及ぶこととなる。教師信号が毎回全体の計算結果に影響してしまうことで、学習が不安定となることを防ぐため、学習の対象となるネットワークの他に、もうひとつ試行のためのネットワークを用意し、一定回数の学習ごとに学習したネットワークを同期する手法を用いることがある。これがTarget Networkである。

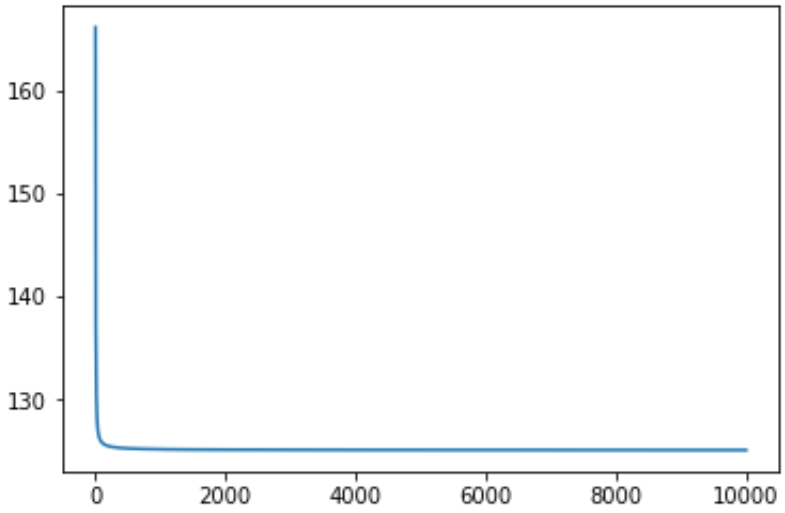
報酬のクリッピング

環境に対する行動によって得られる報酬を1.0から-1.0といった決まった範囲の値に設定する手法を報酬のクリッピングという。ハイパーパラメータの変更によって学習が不安定となることを防止できる。

結果

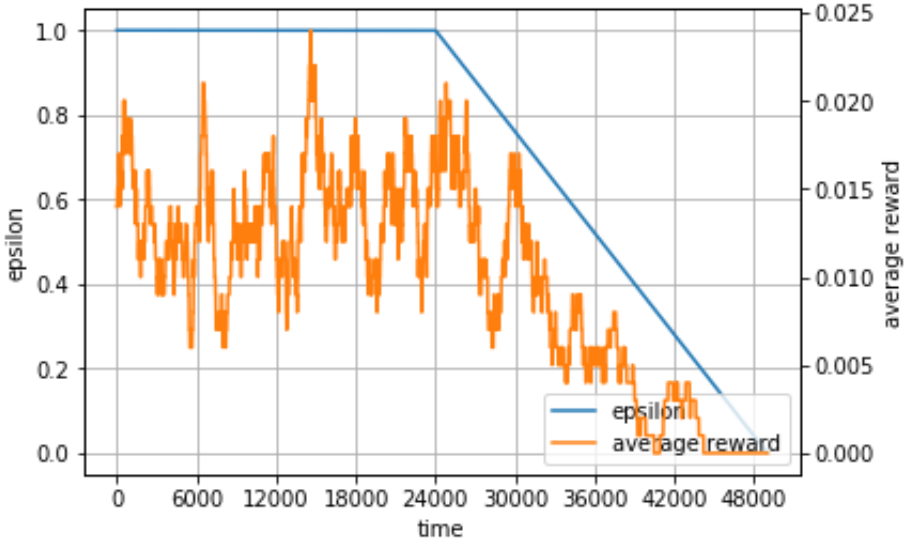
以下のように、OpenAIGymの課題に関しては、十分な収束が得られなかった。

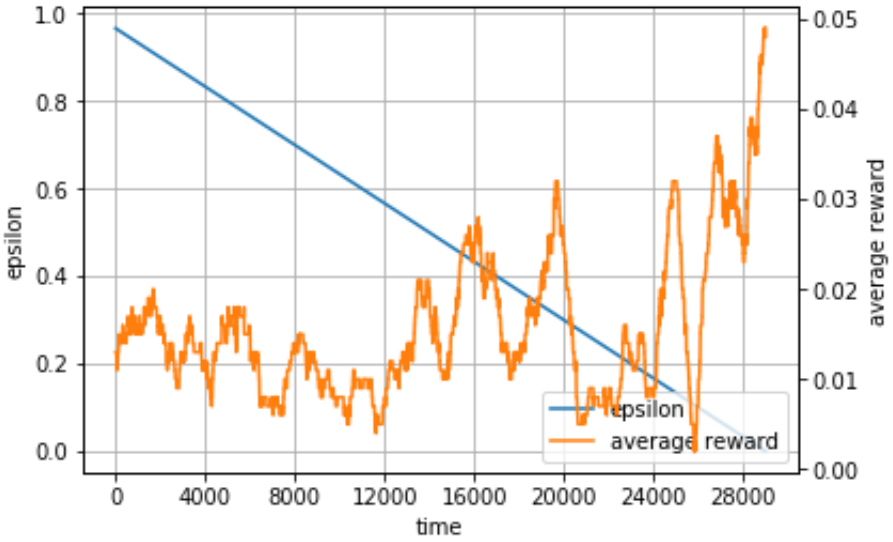
XOR関数のシミュレート



FrozenLake-v0

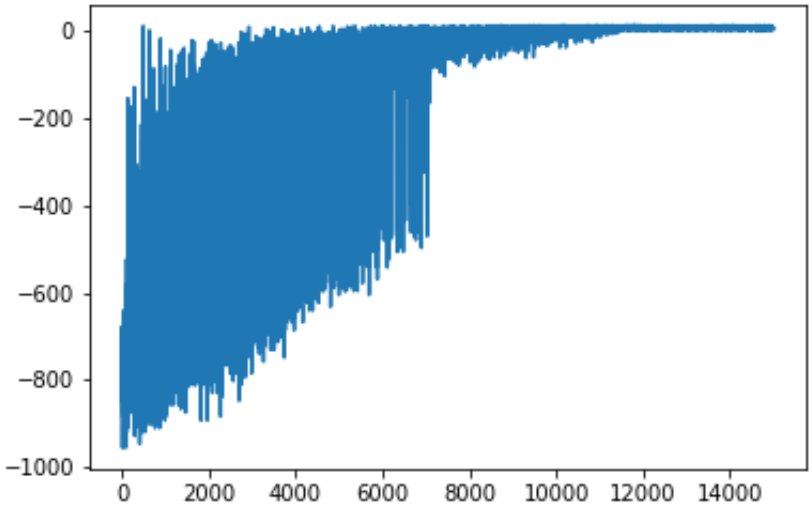
調べたところ、FrozenLake-v0課題では、収束したというためにはおよそ1.0程度の報酬が合計で得られている必要がある。

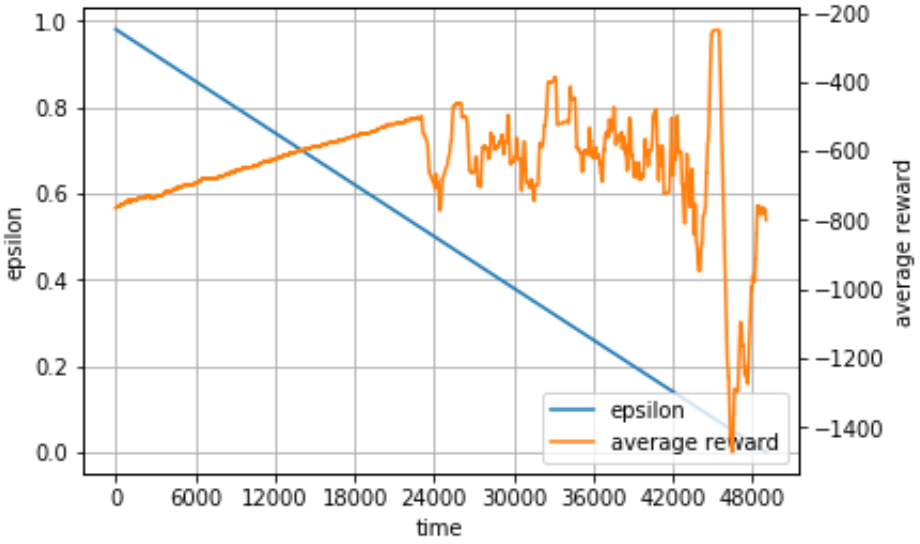


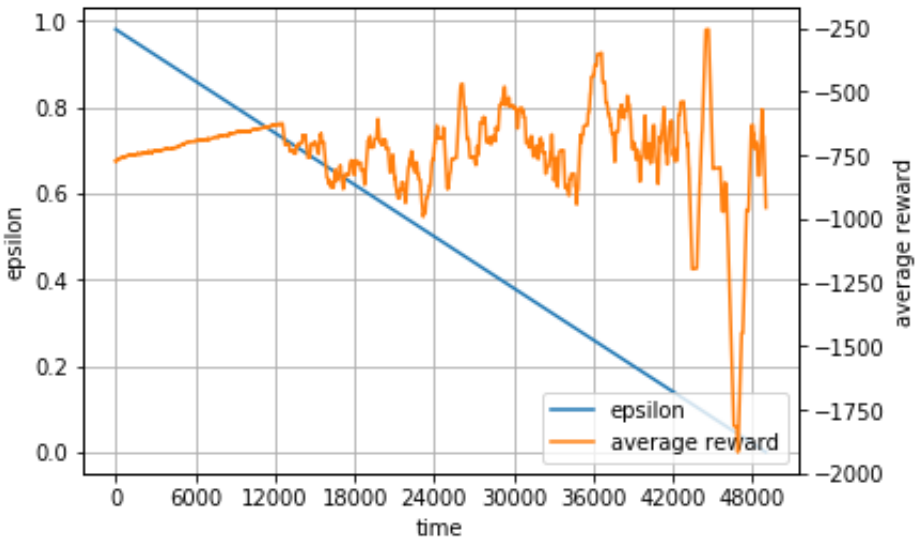


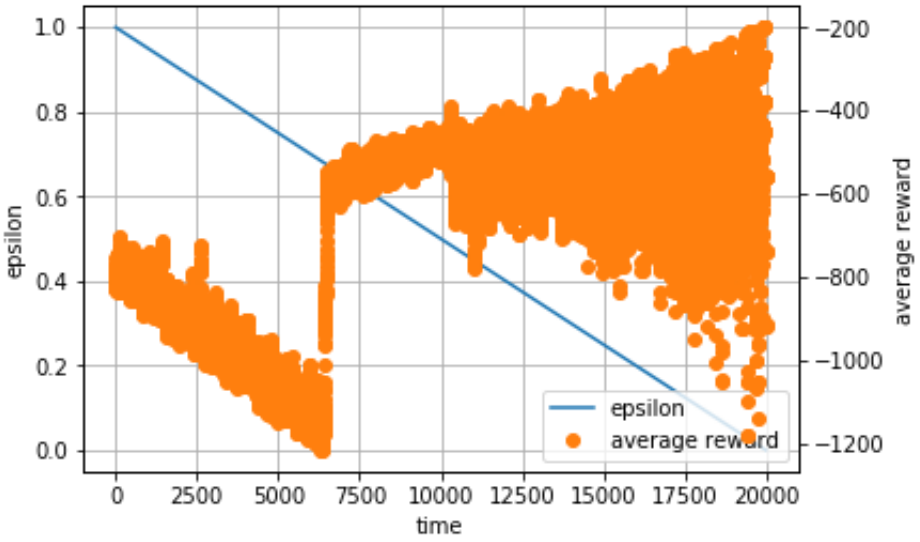
Taxi-v2

一つ目の画像は、以前Q学習によってこの課題を行った際に、収束した学習データである。最終的に正の値の報酬を得ることができれば、学習が収束したと言える。









考察

試行回数の問題

以前に強化学習の一手法であるSarsa学習を実装した際も、Taxi-v2の課題で同じように、合計報酬が-200程度で止まってしまう問題が起きたことがあった。収束が保証されている手法でも同じような現象が起きるのだから、今回も学習の試行回数が少なかっただけで、より時間をかければ収束が見込めた可能性もある。しかしながら、下記のようないくつかの現象から、実装にいくつかの問題があったと考えるのが妥当かもしれない。

正則化を実行するタイミングの問題

今回の実装では、最初のうちは正則化も行うようにしていたが、正則化項を足し合わせる場所に問題があったかもしれない。正則化項は、今回はシグモイド関数の適用の後に足し合わせていたが、これをシグモイド関数の内部におくことで、出力が大きくなることを防止できたかもしれないからだ。実験を行った Jupyter Notebook を見ると、学習を行ったセルの下にはしばしば Warning が発生している旨の出力が見られる。この Warning はほとんどオーバーフローの発生を示唆するものであり、正則化が正しく実行され、シグモイド関数の適用のタイミングが適切であれば、起きづらい問題であったと考えられる。

XOR関数の収束の際のピクつきや勾配確認のズレの問題

今回実装したネットワークのテストにはXOR関数のシミュレートや数値解析との比較による勾配確認を用いたが、これらの結果があまりおもわしくなかったのも気がかりだった。XOR関数のシミュレートに関しては、たびたび学習の際に誤差が波打って収束しないことがあり、有限の試行回数内で必ず振動せず収束するという保証はないとはいえ、単純な関数のシミュレートに対して綺麗な収束が得られないのは問題だった。数値解析に関しては、多数の入力に対して一度に計算を行っているため、どの程度の誤差であれば許容できるのかが判断しづらかったが、一般的な値と比べるとかなり大きな誤差があった。これらの問題は、ネットワークの実装、特に勾配の計算に問題がある可能性を示唆していると言える。

非効率な実験に関する定性的な考察

最後に、実習全体を通しての問題として、実験の繰り返しが非常に非効率であった。 Jupyter Notebook を用いた作業では、Pythonのプロセスは基本的に逐次計算となり、複数の実験を同時に行うことができないだけでなく、計算自体が非常に実行時間のかかるものだったため、問題に対して解決策を試せる回数が少なくなってしまった。この問題に関しては、コードをPythonのスクリプトに移し、いくつかのパラメータや、どの手法を適用するかしないかなどの選択肢をコマンドライン引数にして渡せるようにしていれば解決できたと考えられる。