Preferred Networks インターン選考 2017 コーディング課題機械学習・数理分野

変更履歴

- 2017 年 5 月 12 日：初版

回答にあたっての注意

- 課題 1-5 では、各言語の標準ライブラリの関数のみを使用し、NumPy, Eigen など多次元配列ライブラリは使用しないで実装してください。
- 課題 6 ではライブラリを自由に使用して頂いて構いません。
- プログラムの回答には以下のいずれかの言語を利用してください。
- C, C++, Python, Ruby, Go, Java
- 本課題では pipe/fork を用いるコードを実行する必要があります。MacOS/Linux では特に問題ないはずです。Windows の場合は cygwin や msys2 で動作確認をしています。

提出物

- 課題 1-6 のプログラム、課題 5 のパラメータをダンプしたファイル、課題 6 のレポートをそれぞれ提出してください。プログラムは課題ごとに別々になっていてもよいですし、1 つのファイルにまとまっていても構いません。
- プログラムはできるだけ他人が読んでも分かりやすいものになっており、また追試がしやすい形になっていることが望ましく、レポートはわかりやすくまとまっているのが望ましいです。
- プログラムのビルドに必要な環境・実行環境・実行手順について記述してください。
- プログラムの説明を記述した補足資料（A4 用紙 1 枚以内）を添付しても構いません。

提出方法

「事前課題の提出物」については Google drive にアップロードの上、共有 URL を下記の応募フォームに記入してください。アップロード手順は以下の URL をご参照ください。

- 応募フォーム：https://docs.google.com/a/preferred.jp/forms/d/e/1FAIpQLSd_zC_XT2dHMyRO9WQ-YuRU0sx2HeQIep-NBoqMWpN_j8KNw/viewform
- アップロード手順：https://www.preferred-networks.jp/wp-content/uploads/2017/04/intern2017_GoogleUpload_3.pdf

問い合わせ

課題に関する質問などは intern2017@preferred.jp にお願いします（応募時と同一のメールアドレスです）

問題文

本課題では、強化学習のタスクに最適化アルゴリズムである Cross Entropy Method (CEM) を適応してエージェントを学習させることを目標にします。強化学習とは**エージェント**と呼ばれる意思決定をするアルゴリズムが**環境**とのインタラクションを通じて良い方策を求める学習手法のことです。エージェントは環境から与えられる観測情報に基づいて行動を起こし、環境はエージェントの行動を受けて内部状態を更新した後、現在の観測情報と**報酬**と呼ばれる値を返します。報酬はエージェントの取った行動がどの程度良かったのかを表します。1 回の行動は**ステップ**と呼ばれることがあります。環境はあるときにインタラクションの終了信号を伝えることがあります。エージェントが行動をスタートしてから環境から終了信号が伝えられるまでの一連の流れを**エピソード**と呼びます。エピソード内の報酬の合計値を**収益**と呼びます。強化学習アルゴリズムの目標はエージェントができるだけ多くの収益を得られるように方策を改善することです。

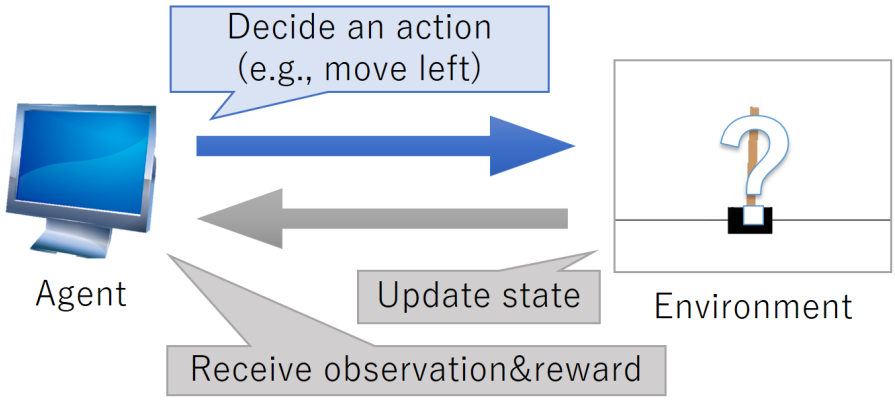

Figure 1:

本課題ではカートポールと呼ばれる古典的な制御系タスクを解いていただきます。課題全体を通じて、プログラム内では環境を 1 つのクラスで表すものとします (クラスの無い言語を使う場合はクラスに相当する関数群などで環境を表して下さい)。環境は以下の機能を持つメソッドをインターフェースとして実装しているものとします。なお、これは OpenAI Gym で実装されているインターフェースを模倣したものです。

- `reset()`: 環境を初期化するメソッドです。引数はありません。返り値には環境の初期状態の観測情報を表すベクトルが来ます。

- `obs_dim()`: 観測情報を表すベクトルの次元を返すメソッドです。
- `step(action)`: 環境に対して行動を起こすメソッドです。引数として行動を表す値 `action` を取ります。本課題では簡単のために、`action` は常に 1 か -1 のどちらかであるとします。環境は行動を受けて内部状態を更新した後に、観測情報を表すベクトル、環境が終了したかどうかを表す bool 値、報酬値の 3 つを返します。

課題 1

まずは検証用に、学習が容易な環境を作ることにします。以下のインターフェースと仕様に沿うような環境を表すクラスを作って下さい。作ったクラスが正しく動くことを確認する簡潔なコード (単体テスト) も書いて下さい。

クラス名：`EasyEnv` メソッド:

- `reset()`: 区間 $[-1, 1]$ から一様にランダムサンプリングした値を 1 次元のベクトルとして返してください。
- `obs_dim()`: ここでは常に 1 を返して下さい。
- `step(action)`: 直前のステップの観測情報 (1 次元ベクトルの要素) を `prev_obs` とします。もし `reset` の直後に `step` が呼び出されていた場合、`prev_obs` は `reset` の返り値を指すものとします。このステップの観測情報は [-1, 1] から一様にランダムサンプリングした値を返してください。報酬は `action * prev_obs` としてください。環境がスタートしてから 10 ステップ後に環境を終了させてください。

課題 2

課題ディレクトリに `cartpole.cc` というファイルがあります。このソースコードはこの課題であるカートポールの問題環境を表しています。例えば以下のようなコマンドでコンパイル出来ます。

```
g++ -std=c++11 cartpole.cc -o cartpole.out
```

ここでコンパイルした `cartpole.out` を便宜上ホストプログラムと呼ぶことにします。ホストプログラムの実行時にはコマンドライン引数にエージェントに相当するプログラムを指定します。ホストプログラムは内部でエージェントに相当するプログラムを起動し、標準入出力を相互に介してやり取りを行います。サンプルプログラムとして、C++ で書かれた `random_action.cc` と、python3 で書かれた `random_action.py` というコードがあります。以下のようなコマンドでこれをエージェントとして実行できます。

```
# python3
./cartpole.out "python3 random_action.py"
# C++
g++ -std=c++11 random_action.cc -o random_action.out
./cartpole.out ./random_action.out
```

エージェントに相当するプログラムがホストプログラムとやりとりをするためには以下のいずれかの文字列を標準出力に送る必要があります。

- 標準出力に `r` と出力して flush する：環境を初期化した後に、初期の観測情報が標準入力を介して送られます。形式は `obs x1 x2 x3 x4` となっています。ここで、各 `xi` は実数値です。
- 標準出力に `s <action>` と出力 (ここで`<action>`は 1 か-1 のどちらか) して flush する：環境に対して行動を起こします。この出力を送ると環境の内部状態が更新され、初期化時と同じ形式で観測情報が標準入力を介して送られます。もしポールが一定の角度を超えた場合、環境は停止してしまい、観測情報の代わりに `done` という文字列が送られます。
- 標準出力に `q` と出力して flush する：ホストプログラムを終了させます。このコマンドを送らずにエージェントに相当するプログラムを終了させると、ホストプログラムがエラーを出す可能性があるため注意してください。

標準入出力による通信部分をラップするようなクラス `CartPoleEnv` を書いて下さい。`CartPoleEnv` は課題 1 と同様のインターフェース (`reset`, `obs_dim`, `step`) を持ち、また以下の仕様に沿っているものとします。作ったクラスが正しく動くことを確認する簡潔なコード (単体テスト) も書いて下さい。

- 報酬は常に 1 を返すものとします。
- 環境は 500 ステップで停止させてください。

なお、デバッグメッセージなどをコンソールに書き出したい場合は、標準エラー出力を使用することをお勧めします。

課題 3

エージェントの方策は何らかのパラメータのモデルで表現されます。本課題では、方策として線形関数をモデルとして用いることにします。以下のようなクラスを実装してください。作ったクラスが正しく動くことを確認する簡潔なコード (単体テスト) も書いて下さい。

クラス名：`LinearModel` メソッド：

- `<コンストラクタ>(initial-param)`: モデルのパラメータをベクトル `initial-param` で初期化します。ここで、`initial-param` の次元は環境が返す `obs_dim()` と同じであるとします。
- `action(obs)`: 観測情報とモデルのパラメータの内積を取り、正ならば 1、そうでないならば-1 を返します。

課題 4

Cross Entropy Method (CEM) は最適化アルゴリズムであり、強化学習で用いる場合は以下のような処理で方策を改善していきます。

- 初期の方策のパラメータを適当に決める。
- 収益が収束するまで以下を繰り返す:

- 今の方策のパラメータを θ とする。
- $i = 1, ..., N$ について以下を行う：ノイズを表すベクトル ε_i (次元は θ と同じ) をサンプリングして、新しいパラメータ $\theta_i = \theta + \varepsilon_i$ をつくる。
- $i = 1, ..., N$ について以下を行う：環境を初期化して、θ_i を方策のパラメータとしてエージェントを 1 エピソード分動かし、得られた収益を計算する。
- 収益の上位 $100\rho\%$ を出した θ_i の平均値を計算し、それを新たな方策のパラメータ θ として更新する。

ここで、N と ρ は事前に決定されているハイパーパラメータです。課題 4,5 では $N = 100, \rho = 0.1$ などを用いて下さい。 また、ノイズベクトル ε_i としては、各要素が平均 0 分散 1 の正規分布から独立にランダムサンプリングされたものを用いて下さい。もしプログラミング言語の標準ライブラリで正規分布が実装されていない場合は、$[-1, 1]$ 上の一様分布から 3 回独立にランダムサンプリングして足し合わせたものを正規分布の近似として使用して下さい。CEM を実装し、`EasyEnv` で正しく動作することを確認して下さい。

課題 5

CEM を用いて `CartPoleEnv` でエージェントを学習させて下さい。学習させたモデルを評価する関数を書いてください。100 エピソード連続で実行して、95 以上のエピソードで収益が 500 に達すれば学習ができたものと見なします。学習ができたときの `LinearModel` のパラメータを以下の形式でファイルに出力して、提出して下さい。ここで、`wi` は `i` 番目のパラメータ (実数値) です。

```
w1
w2
w3
w4
```

課題 6

課題 5 のパラメータや実装を変えたときに実験結果がどのようになるかレポートで報告して下さい。レポートは A4 用紙 1 枚から 2 枚（図・表などを含む）に収めてください。

変更する項目としては例えば以下のようなものがあります。

- 強化学習においてサンプル効率 (方策が収束するまでに必要となるエピソード数やステップ数) はアルゴリズムの良し悪しを測る重要な指標の一つです。N や ρ、ノイズの入れ方を変えた場合にサンプル効率がどのように変わるかを確認して下さい。
- 部分観測性 (環境の状態がすべて観測されないこと) は実機のタスクでよく見られます。`cartpole.cc` で得られる観測情報のうちの一部を除いても学習がうまく進むかを確認して下さい。うまくいかない場合、観測可能な情報から特徴量などを増やすことで学習可能になるか確認して下さい。

- 観測情報にノイズが乗っていたり、また観測情報が数ステップ遅延してやってくることは実機のタスクでよく見られます。これらの要素が学習にどのような影響を及ぼすかを確認して下さい。
- 方策のモデルを線形のようなパラメータの個数が少ないモデルではなく、ニューラルネットのように多くのパラメータからなるモデルに変えた場合に、サンプル効率などがどう変化するか確認して下さい。

これらは一例であり、全てを実施する必要はありません。時間が足りない場合は重要だと思われる項目を実験してください。また、これら以外のものを変更しても構いません。

(課題文ここまで)