Deep Reinforcement Learning with Attention for Slate Markov Decision Processes with High-Dimensional States and Actions

Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold
Google Deep Mind

Abstract

現実の多くの問題では、特徴ベクトルとして表現される行動空間を伴う。高次元制御は非常に解くのが困難な問題ではあるが、近年、適当な次元数においては進展を遂げている。ここで我々は、ある特定の形で、2000 もの高次元な問題について扱った際に成功した試みについて報告をする。標準的な強化学習のフレームワークに当てはまらないようなレコメンドシステムといった重要な応用への適用を動機として、「候補者名簿マルコフ決定過程 (Slate-Markov Decision Processes)」を導入する。Slate-MDP は MDP をもとに実行される一つの行動を持つ元となる行動群候補者名簿から構成される組み合わせによる行動空間を持つ MDP である。エージェントはこの実行される行動の決定を制御することができず、また行動は候補者名簿から選ばれるとは限らない。e.g.,レコメンドシステムにおいては全てのレコメンド候補が選ばれなかったような状態に対応する。全体の候補者名簿の価値を学習するために、状態と行動の両方の特徴表現をもとに Deep Q-learning を使用する。既存の手法とは異なり、我々のタスクの組み合わせ的（combinatorial）逐次的（sequential）の両方の側面で最適化を行う。組み合わせ的で逐次的な長期的な価値という側面を考慮しないような今までのエージェントよりも、新しいエージェントが勝っていることが、現実世界のレコメンドシステムのような動的な様々な問題に対して実証される。さらには、候補者名簿のそれぞれの位置(position)に対して価値が最も高いような行動空間の一部へと注意を向け、このエリアにある行動のみを評価する方策(policy)を学習するために deep な決定的な方策勾配を用いる。その注意(attention)はサブモジュール性(submodularity)を最大限活用するような逐次的でグリーディーな手順の中で利用される。最後に、リスク見いだし(risk-seeking)を行うことで劇的にエージェントのパフォーマンスやより遠くにある戦略の発見を行う能力を向上させることを示す。

1　Introduction

強化学習は未知の環境に対して相互作用を行いながら試行錯誤を通して学習を行うというパラダイムの一つである。その相互作用はエージェントが行動を選択し、環境が実際の報酬と共に観測値を返すというサイクルの中で生じる。エージェントのゴールは長期的な累積報酬を最大化させることである。強化学習は自動ヘリコプター制御や近年では Atari の様々なゲームの攻略、物理制御のタスクなどにおいてなど、数多くの成功事例がある。

これらは、印象的な成果ではあるが、Atari のゲームは高々18 つの行動しかもたず、また物理制御問題は連続な行動空間ではあるが、それらは 10 次元を下回るような次元数にとどまっている。我々の研究では、2000 にまで及ぶ次元の特徴ベクトルにより表現されるような、組み合わせ的行動空間で扱う。強化学習の応用をレコメンドシステムのような問題に対して適用することを想定しており、その中では行動の全体の候補者名簿はそれぞれの時間において選ばれる。これらの問題が組み合わせ的な行動空間を伴う MDP とモデル化され、その応用の中で自然と現れる追加的な構造により、候補者名簿の近似的価値最大化を解く事ができるようになる。

Slate Markov Decision Process (Figure 1)

それぞれの時刻において、エージェントがある有限の集合 A から決められた数の行動を選択するような強化学習の問題について扱う。行動の集合を候補者名簿(slate)と呼ぶことにする。候補者名簿は我々の定式化の中では順番を持つことになるが、特別な場合として、その順番とは無関係な環境を持ちうることもありうる。我々が扱う環境の中では、候補者名簿の中のたった一つの行動のみが実行される。例えば、レコメンドシステムにおいて、レコメンドが与えられた際のユーザーの選択が行動の行使に相当する。伝統的(traditional)な強化学習の問題を基礎として、e.g.マルコフ決定過程(MDP)それを仮定する。

MDP の中では、それぞれの時間 t において行動 s_t が観測され、行動 a_t が行使される。受け取られる次の状態 s_t+1 と報酬 r_t は時刻 t 以前に行われた事象とは独立している。すなわち、s_t は時刻 t までの全ての関連する情報を集約していると想定できる。

Slate-MDP の重要な点は一つの行動を行う代わりに、エージェントは行動の一つの候補名簿全体を選択し、環境はその元となる MDP においてどれが行使さ

れるかを選ぶ、この様子が Figure.1 に図示されている。

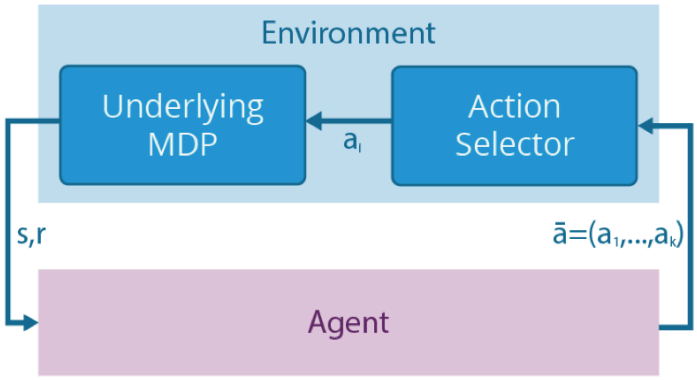

Figure 1: Slate-MDP Agent-Environment Framework.

受け取られる候補者名簿の情報はエージェントにどの行動が実際に行われたかを伝えるが、その他の行動郡については何も情報は伝えられない。Slate-MDP は全ての行動郡が行使され、それぞれの full slate が広大な行動空間の中のそれ自身の離散的な一つの行動であるという状況に比べ、重要な追加的構造を持っている。

Full-slate に対する Slate-MDP の価値関数を直接学習するようなモデルフリーなエージェントについて調べてみる。大規模の応用でしばしばとられている単純なアプローチとしては、それぞれの行動価値を学習して、最も良い行動を結合させる、ということがあげられる。タスクの組み合わせ的な側面を全く考慮しないこの単純な方法の重大な欠点を実験でしめす。追加的な行動郡が一つの候補者名簿に追加される際、これらは最も価値の高い行動価値の行使を妨げることがありうる。候補者名簿価値関数を学習するエージェントは、これによる被害を受けることは少なく、原理的には、information retrieval における最大マージン relevance のような手法とは異なり、数学的な定義や導入すべき多様性の度合いを調整する一定値が与えられれば、“多様性”のような有益な候補者名簿パターンを学習することができる。

Full slate のエージェントの主な欠点としては、それらに基づく候補者名簿をつくるのに必要な価値関数を評価する数が膨大であることである．それゆえ、

価値関数が高いような行動空間におけるある領域にたいしての注意をむけるために決定的な方策勾配を用いてパラメータ化された方策の学習の選択の調査もまた行った。ニューラルネットワークの方策は最近傍法の参照やこの制限された集合における価値関数の評価と合わせて用いられる。

Related Work

割愛

2 Reinforcement Learning with Slate Actions

(p4. l 1~l 27 は一般の強化学習の表記などについての文章であるため割愛)

Slate Markov Decision Processes (slate-MDPs)

このセクションでは、より効率てきな予測を可能にする重要で特殊なケースと同じように slate-MDP を正式に導入する。

Definition 1 (slate-MDP)

Definition 1 (slate-MDP). *Let* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ *be an MDP. Let* $\varphi \colon \mathcal{S} \times \mathcal{A}^l \to \mathcal{A}$. *Define* $\mathcal{T}' \colon \mathcal{S} \times \mathcal{A}^l \times \mathcal{S} \to \mathcal{P}(\mathcal{S})$ *and* $\mathcal{R}' \colon \mathcal{S} \times \mathcal{A}^l \times \mathcal{S} \to \mathcal{P}(\mathbb{R})$ *by*

$$\mathcal{T}'(s, \mathbf{a}, s') = \mathcal{T}(s, \varphi(\mathbf{a}), s'), \mathcal{R}'(s, \mathbf{a}, s') = \mathcal{R}(s, \varphi(\mathbf{a}), s').$$

The tuple $\langle \mathcal{S}, \mathcal{A}^l, \mathcal{T}', \mathcal{R}' \rangle$ *is called a* slate-MDP *with* underlying MDP $\mathcal{M}$ *and* action-execution function φ. *We assume that the previous executed action can be derived from the state through a function* $\psi \colon \mathcal{S} \to \mathcal{A}$.

Note that any slate-MDP is itself an MDP with a special structure. In particular, the probability distribution of the next state s' and the reward r' conditional on the current state s and action $\mathbf{a}$ can be factored as:

$$\underbrace{\Pr(s', r' | s, \mathbf{a})}_{\mathcal{R}' \circ \mathcal{T}'} = \sum_{a \in \mathcal{A}} \underbrace{\Pr(s', r' | s, a)}_{\mathcal{R} \circ \mathcal{T}} \underbrace{\Pr(a | s, \mathbf{a})}_{\varphi}.$$

The expected reward for a slate-MDP can be computed as

$$\bar{\mathcal{R}}(s, \mathbf{a}) = \sum_{a \in \mathcal{A}, s' \in \mathcal{S}} \Pr(s' \mid s, a) \Pr(a \mid s, \mathbf{a}) \bar{\mathcal{R}}(s, a, s')$$

If we let $Q^{\text{exec}}(s, a, s') = \Pr(s' \mid s, a)(\bar{\mathcal{R}}(s, a, s') + \gamma V^{\pi}(s'))$, we have the following identity for the state-slate value function of the slate-MDP:

$$Q^{\pi}(s, \mathbf{a}) = \sum_{s' \in \mathcal{S}, a \in \mathcal{A}} \Pr(a \mid s, \mathbf{a}) Q^{\text{exec}}(s, a, s') \tag{1}$$

for any slate policy $\pi : \mathcal{S} \to \mathcal{A}^l$.

ここで、行使される行動 ϕ (s_(t+1))は a_t の要素の一つであることを要求はしないが、その環境の中で、「良い」候補者名簿 a in A^l を作成する（Definition 2 における Execution-Is-Best)。レコメンドシステムの設定では、ϕ (s_(t+1)) in a_t はそのレコメンドがユーザーによって選ばれたことを意味する。候補者名簿から行使される行動を持っていることが最も良い結果であると形式的に定義することにする。方策間の価値の順序(value-order)は報酬ゼロでエピドードが終了することをほのめかすような、ϕ (s_(t+1)) not in a_t であるような環境を修正したときのそれと一致する。

Definition 2 (Value-order, Fatal Failure, Execution-Is-Bdst(EIB))

Definition 2 (Value-order, Fatal Failure, Execution-Is-Best (EIB)). *Let μ and ν be two environments with the same state space $\mathcal{S}$ and action space $\mathcal{A}$.*
Value order: *If, for any pair of policies π and $\tilde{\pi}$,*

$$V_{\mu}^{\pi}(s) \geq V_{\mu}^{\tilde{\pi}}(s) \iff V_{\nu}^{\pi}(s) \geq V_{\nu}^{\tilde{\pi}}(s) \; \forall s,$$

then we say that μ and ν have the same value-order.

Further, suppose there is $s_{\text{end}} \in \mathcal{S}$ such that $\nu(s_{\text{end}}, r' = 0 \mid s_{\text{end}}, \mathbf{a}) = 1$.
Fatal Failure: *If $\nu(s_{\text{end}}, r' = 0 \mid s, \mathbf{a}) = 1$ whenever $\psi(s') \notin \mathbf{a}$, then we say that ν has fatal failure.*

Suppose that $\nu(s', r' \mid s, \mathbf{a}) = \mu(s', r' \mid s, \mathbf{a})$ if $\psi(s') \in \mathbf{a}$, i.e., the environments coincide for executed slates.
EIB: *If ν has fatal failure and μ has the same value-order as ν, then we say that μ has Execution-Is-Best (EIB) property.*

大規模な状況下で候補者名簿の価値の最大化を定められるようにするために、組み合わせ的な探索をさける必要がある。まずはじめに、もし環

境が fatal failure の特性を持っている場合はΣ_(a in A)は(1)における
Σ_(a in a)によって置き換えられることに留意する。言い換えれば、候補者名簿の中の行動に対応する項のみが非ゼロである。この条件は我々の環境では成り立たないため、EIB の仮定が自然であり、Definition 2 で定義されるような修正した環境ν について学習を行うことを暗に示している。より少ない項についての和であれば最適化がしやすいが、問題は依然として組み合わせ的でなおかつスケールしないことにある。それゆえ、もし f:U_(j=0)^l A^j → R が単調性でサブモジュラであれば、グリーディーに候補者名簿 a_(greed)を逐次的に選択すると f(a_(greed)) >= (1 − 1/e)max_a f(a)となるため、我々にとって単調性(monotonic)とサブモジュラリティ(submodularity)は興味深い。

Definition 3 (Monotonic and Submodular)

Definition 3 (Monotonic and Submodular). *We say that a function* $f : \cup_{j=0}^{l} \mathcal{X} \to \mathbb{R}$ *for* $\mathcal{X} \subset \mathcal{A}^l$ *is*
Monotonic *if* $\forall a, a_1, \ldots, a_i \in \mathcal{A}$ *it holds that* $f((a_1, \ldots, a_i, a)) \geq f((a_1, \ldots, a_i))$ *and*
Submodular *if (diminishing returns)*

$$f((a_1, \ldots, a_i, a)) - f((a_1, \ldots, a_i)) \leq f((a_1, \ldots, a_{i-1}, a)) - f((a_1, \ldots, a_{i-1}))$$

holds for all $a, a_1, \ldots, a_i \in \mathcal{A}$.

単調性とサブモジュラリティを保証するために、行動選択が環境において逐次的に起こる場合（e.g.レコメンドがあるユーザーに対して１つ１つ提示されるか、もしくはユーザーがそのような様式で閲覧するといった場合）に成立するので逐次提示(sequential presentation)と呼ぶさらなる仮定を導入する。我々が行った環境が逐次提示を持たないが、逐次的にグリーディーな手順はうまく機能する。最初のレコメンドの選択を評価する際は、デフォルトの戦略によって与えられるその他のレコメンドの中にそれを見る。このことは我々の環境設定をより逐次的なレコメンドに近づけるだろう。

Definition 4 (Sequential Presentation)

Definition 4 (Sequential Presentation). *We say that a slate-MDP has sequential presentation if for all states s its action-execution probabilities satisfy*

$$\Pr(a|s,(a_1,\ldots,a_i,a,a_{i+1},\ldots)) = \Pr(a|s,(a_1,\ldots,a_i,a)) \tag{2}$$

and $\Pr(a|\, s,(a_1,\ldots,a_i,a)) \leq \Pr(a|\, s,(a_1,\ldots,a_{i-1},a))$.

Proposition 1. *If a slate-MDP has sequential presentation and satisfies the fatal failure property then its state-slate value function Q^π is monotonic and submodular for all π.*

Proof. Let $\mathbf{a}_k = (a_1,\ldots,a_k)$. For any vector $\mathbf{a}$ and any scalar a, let $\mathbf{a}a$ denote the vector constructed by concatenating a to $\mathbf{a}$. Assume that $a \notin \mathbf{a}_i$. Also the rewards are nonnegative. Then, we have that $Q(s,\mathbf{a}_i a) - Q(s,\mathbf{a}_i) =$

$$\sum_{s'\in\mathcal{S}a'\in\mathbf{a}_i a} \Pr(a'|s,\mathbf{a}_i a)Q^{\text{exec}}(s,a',s') - \sum_{s'\in\mathcal{S}a'\in\mathbf{a}_i} \Pr(a'|s,\mathbf{a}_i)Q^{\text{exec}}(s,a',s') =$$

$$\sum_{s'\in\mathcal{S}a'\in\mathbf{a}_i} \Big[\Pr(a'|s,\mathbf{a}_i a) - \Pr(a'|s,\mathbf{a}_i)\Big] Q^{\text{exec}}(s,a',s') + \Pr(a|s,\mathbf{a}_i a)Q^{\text{exec}}(s,a,s') =$$

$$\Pr(a|s,\mathbf{a}_i a)Q^{\text{exec}}(s,a,s').$$

Sequential presentation immediately implies that $\Pr(a|s,\mathbf{a}_i a) \leq \Pr(a|s,\mathbf{a}_{i-1}a)$. This establishes that $Q(s,\mathbf{a})$ is indeed submodular in $\mathbf{a}$. Monotonicity follows from (2). □

次の節では、この節の理論に基づいたエージェントを導入する。それらは full slate の価値を学習し、EIB と共に逐次提示であるという仮定の元で、組み合わせ的な探索よりも潜在的にはわずかに悪い性能を示す、逐次的なグリーディーな手順を通して、候補者名簿を選択する。さらには、EIB を動機とし、fatal failure が成り立つような修正された環境において学習は行われる。

Slate agents

それぞれの行動の価値(Algorithm 1)をや full slate の価値(Algorithm 2)を直接学習するモデルフリーのエージェントについて考える。後者にたいしては、上記のスロットの中の行動への依存性のみを考慮するような行動選択を実行した。しかしながら、状態と全ての行動を引数とした特徴量を用いた価値関数近似を用いて全体の候補者名簿に依存する価値関数の学習をおこなった。逐次的にグリーディーな方法で最大化を行い、評価されたのと同じ行動と共に最大化されたものに従うスロットを満たし、その前に採った行動は固定化した状態を保った。Algorithm 1 と Algorithm 2 は一般的な方法で述べた一方で、実験の中では

Algorithm 3 ではいわゆる experience replay や target network といった学習を安定化、高速化する役に立つテクニックを含んでいる。Algorithm 1 は 2 つのフェーズで表現されている； 候補者名簿サイズ 1 を用いて学習を行うフェーズ、候補者名簿サイズ l でテストを行うフェーズである。実験では、学習フェーズ野中にテストフェーズを挟み込んでいる。

Deterministic policy Gradient (DPG) Learning of Slate Policies to Guide Attention

候補者名簿を選択する際に価値関数の評価の数を低減させるため、Algorithm 3 で見られるような、学習された価値関数を最大化するような候補者名簿を作成するように学習された attention guiding policy を学習することを試みる。それは最近傍法として使われた候補行動郡があるような Algorithm 2 の一般化である。その方策はそれらや状態の関数としての Q◯π におけるパラメータの勾配上昇を用いて最適化を行う。

これらの追加的な問題は、現存する決定的な方策勾配の研究[~~]と比較しても、さらに高い次元を伴っており、ニューラルネットワークにより生み出される連続な行動を用いることの代わりとして、離散的な部分集合から選ばなければならない。利用出来る行動の中から k 近傍法参照を実行し、最近傍点を実行するかもしくは、特定された全ての近傍に対して Q を評価し、最も高く評価された行動を選択する。この手法のさらなる詳細や、大規模な行動空間での開発は[DAES16]を参照されたい。方策は単にそれ対して、可能な限り高い Q 値を伴ったベクトルを生み出してもらいたいため、同じ方法で更新することになる。しかしながら、Q を学習する時もまた、実際に行われる行動にたいして Q(s,a) に基づいて更新する。[~~]の中にあるように、TD-error(Q(s,a)-r- γ Q'(s',a'))のために用いられる次の行動は現在のターゲット方策によって生み出される π(s')である。複数の候補者名簿に対して最近傍法を利用するために、ある時間の slot に焦点を当てる。Algorithm2 の中で定義された逐次的なグリーディーな最大化を用いたが、それぞれの slot に対して、その選択肢は参照の結果を考慮することにのみ制限される。

3.Experimental comparison

（割愛）

4.Conclusions

レコメンドのような重要な応用でみられるような高次元の組み合わせ的な候補群行動空間を持つ逐次的な意思決定問題をうまく対処するエージェントを導入した。ここでは候補者名簿 Markov Decision Processe の導入に焦点をあて、そのような応用に対する形式的なフレームワークを提供した。関連するベースラインを超えるような新しいエージェントの優位性は生のレコメンドシステムのような現実世界のデータから得られる様々な問題に対して実証された。

Algorithm 1: Generic Simple (Top-K) Slate Agent

Require: trainSteps, testSteps, update, $\varepsilon \geq 0$, $l \geq 1$

$t = 1$, initialize θ for $Q_\theta(s, a)$
Receive initial state s and take random action a
Receive reward r and state s'.
repeat
  Pick a' ε-greedily (slate size 1) from $Q_\theta(s', \cdot)$
  Update θ using update(s, a, r, s', a')
  $s = s'$, $a = a'$
  Perform action a in environment (slate size 1)
  Receive new state s' and reward r
  $t = t + 1$
until $t \geq$ trainSteps
$t = 1$
Receive initial state s
repeat
  Sort the available actions such that $Q_\theta(s, a_i) \geq Q_\theta(s, a_{i+1})\ \forall i$
  Take slate-action $\mathbf{a} = (a_1, .., a_l)$
  Receive reward r and state s'
  $t = t + 1$, $s = s'$
until $t \geq$ testSteps

Algorithm 2: Generic Full Slate

Require: Steps, update, $\varepsilon \geq 0$, $l \geq 1$

$t = 1$, initialize weights θ for $Q_\theta(s, \bar{a})$
Receive initial state s and take random slate-action $\bar{a}$
Receive reward r and state s'.
repeat
  for i=1,l **do**
    Set $a_i = \arg\max_{a \in \mathcal{A}(s')} Q_\theta(s', a_1, \ldots, a_{i-1}, a, a, \ldots)$
  end for
  $\mathbf{a}' = (a_1, \ldots, a_l)$
  Update θ using update$(s, \mathbf{a}, r, s', \mathbf{a}')$
  $s = s'$, $\mathbf{a} = \mathbf{a}'$
  Perform slate-action $\mathbf{a}$ in environment
  Receive new state s' and reward r
  $t = t + 1$
until $t \geq$ Steps

Algorithm 3: DPG+kNN

Randomly initialize $Q(s, a|\theta^Q)$ and policy $\pi(s|\theta^\pi)$ with weights θ^Q and θ^π.
Initialize target network Q' and π' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\pi'} \leftarrow \theta^\mu$
Initialize replay buffer B
Receive initial observation state s_1
for t = 1, T **do**
    With probability $1 - \varepsilon$ select action a_t as $\arg\max_a Q(s, a|\theta^Q)$ where a ranges across the
    k nearest candidate actions of $\pi(s_t|\theta^\mu)$, and with probability ε a random candidate action. For full
    slate agents, $\arg\max$ is replaced by sequentially greedy maximization as in Algorithm 2.
    Perform a_t, receive reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in B
    Sample (s, a, r, s') from B and choose a' as a_t
    was chosen above but with s', Q' and π'.
    Set $y = r + \gamma Q'(s', a')|\theta^{Q'})$
    Update Q by gradient updates for the loss: $L = (y - Q(s, a|\theta^Q)^2$
    Update the policy π using the sampled gradient:

$$\nabla_{\theta^\pi} Q \circ \pi|_s \approx \nabla_a Q(s, a|\theta^Q)|_{\pi(s)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_s$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$$

$$\theta^{\pi'} \leftarrow \tau\theta^\pi + (1 - \tau)\theta^{\pi'}$$

end for