CEDEC 2020

Brand New!

自己対戦と強化学習による NPCの意思決定の研究事例

株式会社スクウェア・エニックス
テクノロジー推進部　AIエンジニア
眞鍋　和子

CEDEC 2020
Computer Entertainment Developers Conference

チーム紹介

- 眞鍋 和子
  - モンテカルロ木探索
- Leandro Graciá Gil
  - DQN
- Gravot Fabien
  - AIアドバイザー
- 重国 和宏
  - システムインフラ

題材

題材カードゲーム

- ドラゴンクエストライバルズ（以下、DQR）
- ターン制バトルを採用した、オンライン対戦カードゲーム

デッキ・戦術の多彩さ

- リーダーキャラクターごとの特技

  例：

  - テリー：貫通属性を持った攻撃
  - トルネコ：道具カード3枚を手札に加える
- 数か月ごとに、新しいカードパックが追加
  - デッキとリーダーのシナジー

一方で

対戦AIのニーズの高まり

色んなデッキを、じっくり試したい

対人戦は緊張するので、もっと気軽にバトルしたい

カードのうまい使い方がわからない

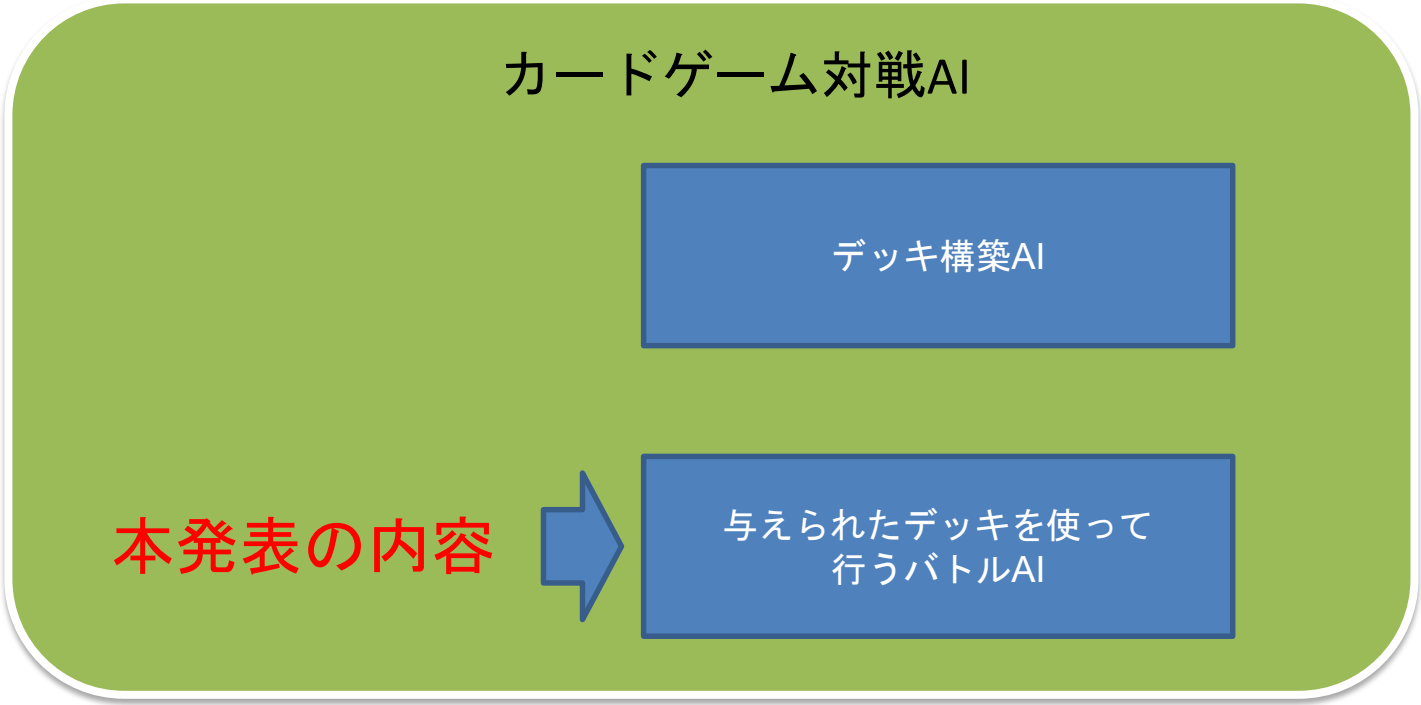
対戦AIの難しさ

- 刻一刻と変わる、環境への対応
  - 新規カード
  - 廃止カード
  - 流行の変化

- 対戦AIの構築手法を研究

2020年8月、DQRに
新しいソロモードによる、
敵NPCとオートプレイが実装されましたが、
本発表の内容と関連はありません。

本研究のターゲット



題材ゲームの分類

不完全情報ゲーム　難易度：激辛

- 一方のプレイヤーからは見えない情報がある

⇒取りうる行動・次の状態の数が膨大なものになる

相手の手札

自分の山札

敵の山札

題材ゲームの分類

不確定ゲーム 難易度：辛口

- ランダム要素を含む
  - バトル開始時、山札がシャッフルされる
  - ランダムな効果を持つカード

⇒状態数の増加

概要

アプローチ

- 1. モンテカルロ木探索
  - リアルタイムでの探索
- 2. DQN
  - 事前学習による評価器

使用デッキ

- デッキ内容は固定
- デッキ内のカードは30枚（1種類につき2枚が上限）
  - テリー: 16種
  - トルネコ: 19種

共通カード（双リーダーが所持）

共通カード

リーダー専用カード

テリー	トルネコ
メラゴースト	メラゴースト
モーモン	モーモン
りゅうき兵	りゅうき兵
ドロル	ドロル
オーガキング	オーガキング
リザードマン	アルミラージ
リビングスタチュー	あらくれチャッピー
テラノライナー	クイーンスライム
バードファイター	スライムダーク
タップペンギー	ビッグハット
デスターキー	つむりんママ
みなごろし	デンデン竜
やいばくだき	はぐれメタル
ギガスラッシュ	ベビーパンサー
ことだまつかい	ドラゴメタル
しっぷう突き	てんばつの杖
	ちからのたね*
	いのちのきのみ*
	しあわせのたね*

*トークンカード。特技によってバトル中に生成される。デッキには含まれない

モンテカルロ木探索

モンテカルロ木探索

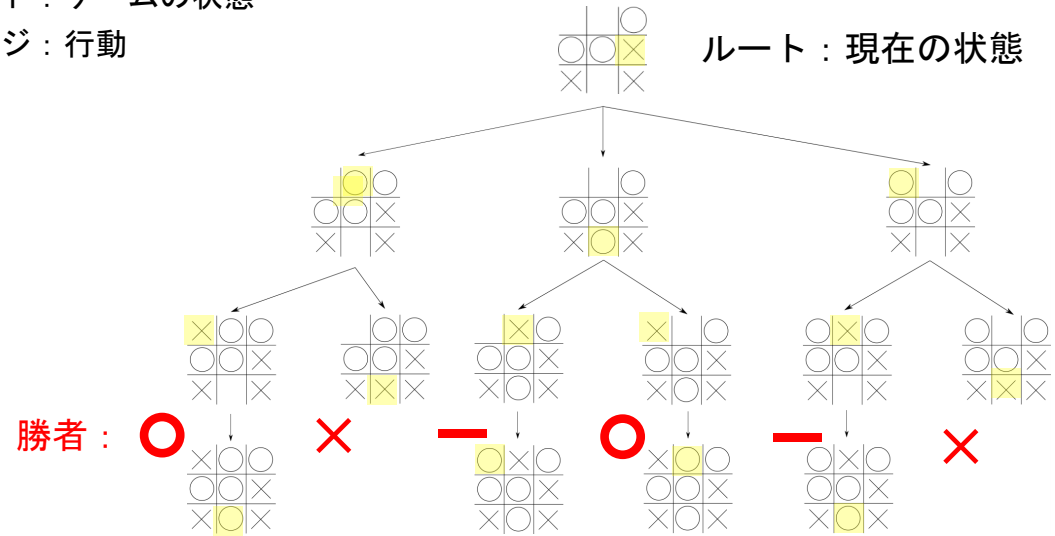
- Monte Carlo Tree Search（MCTS）
- モンテカルロ法を用いた木探索
  - モンテカルロ法
    - 多数のサンプリングにより、確率的モデルに基づいて推定値を得る手法
  - 木探索
    - 状態を木で表現し、最良の評価値を探索する手法

実用事例

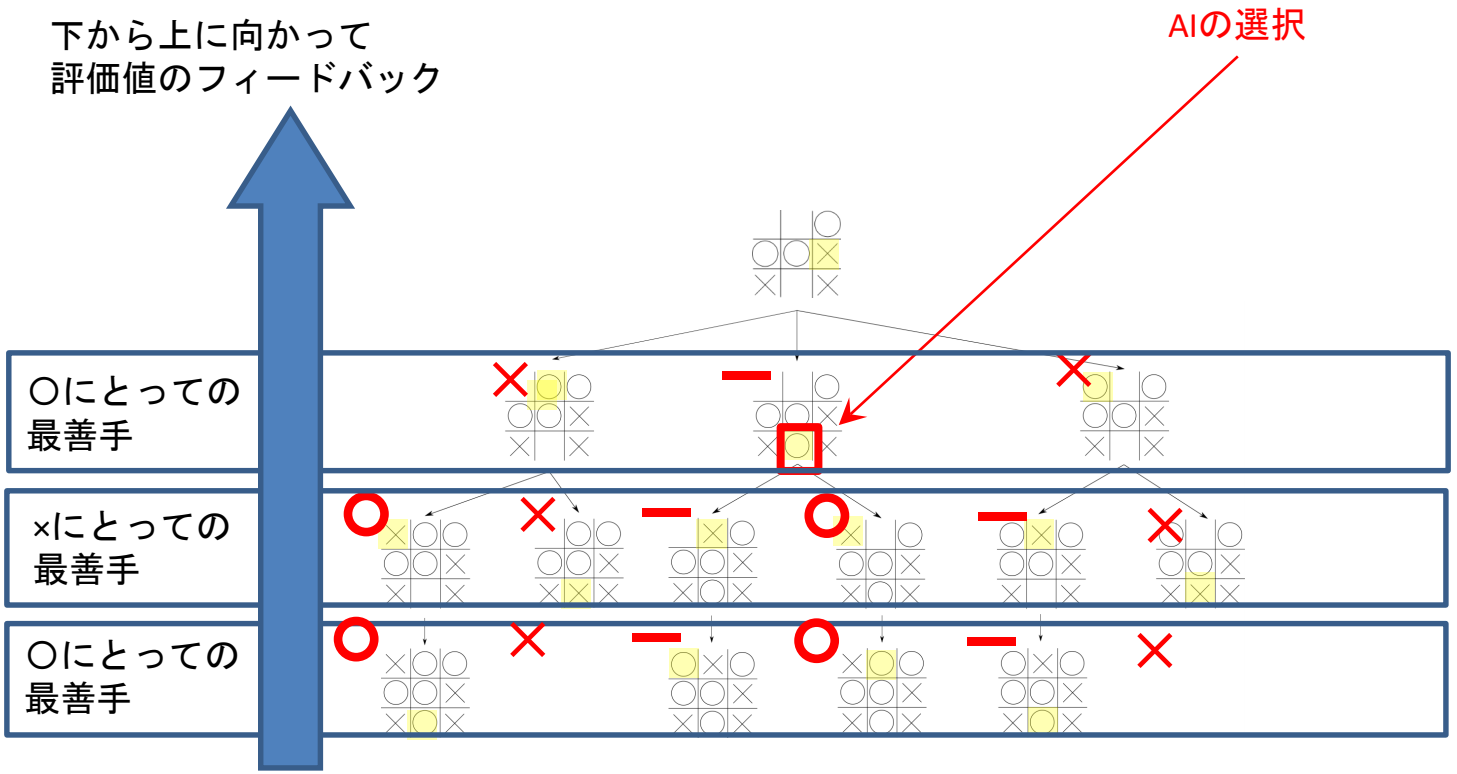
- TOTAL WAR: ROME II
  - Monte-Carlo Tree Search in TOTAL WAR: ROME II’s Campaign AI
    （https://web.archive.org/web/20170313041719/http://aigamedev.com/open/coverage/mcts-rome-ii/）
  - 抽象的な状態をノードとして表現

ゲーム木

- ゲームの状態遷移を表す
  データ構造
- 木構造
  - ノード：ゲームの状態
  - エッジ：行動

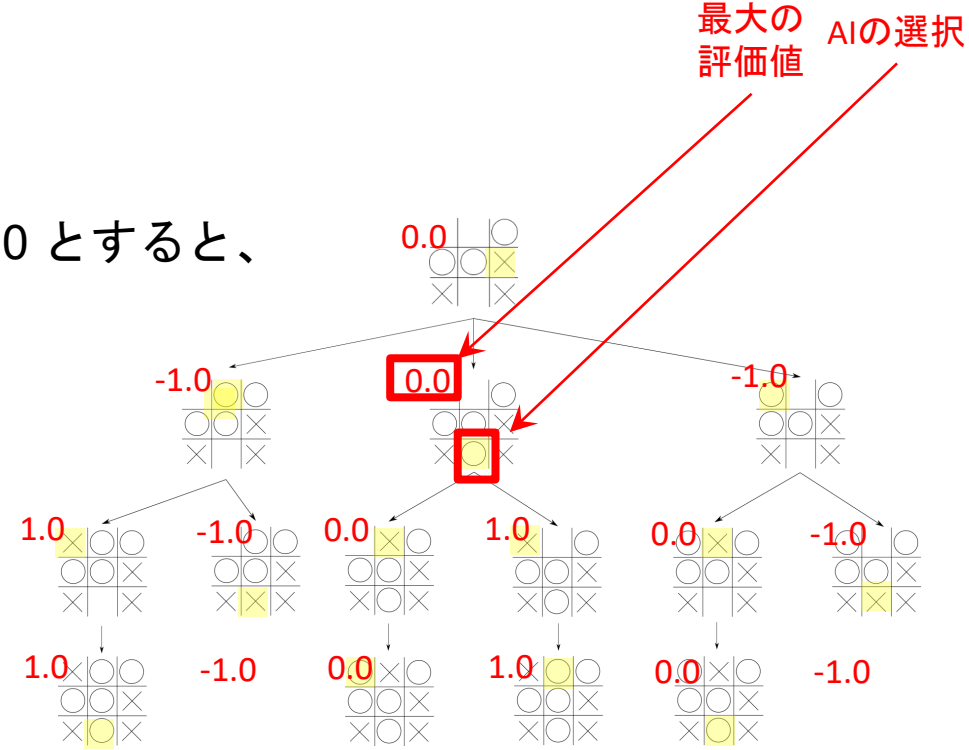


木探索：AIの行動選択



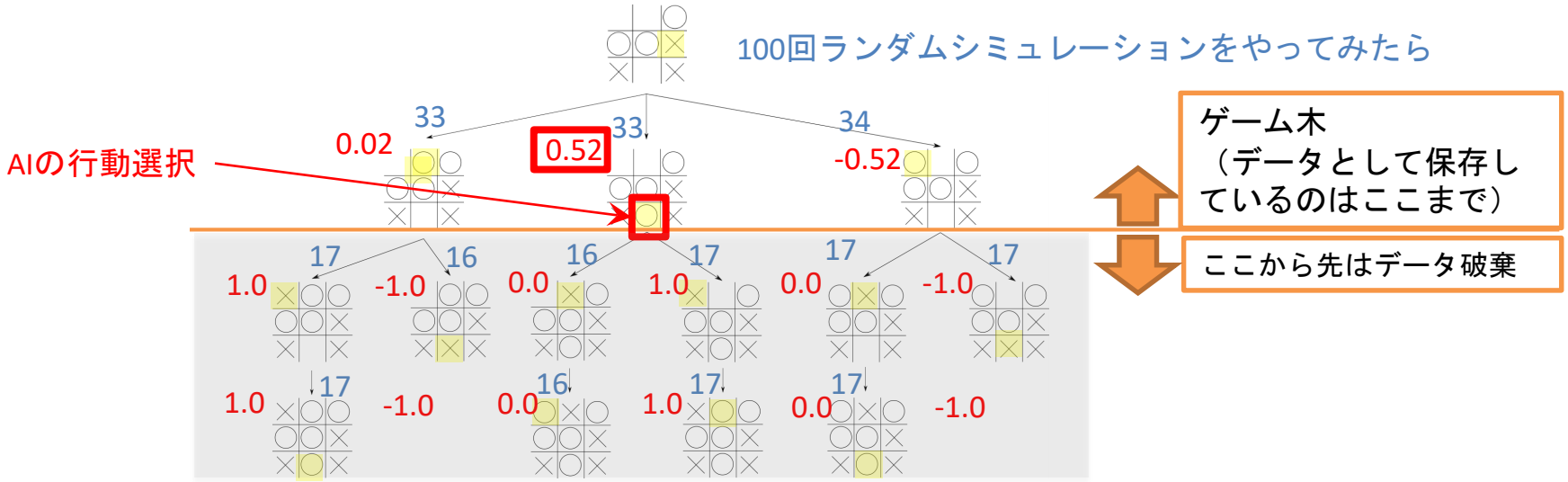
木探索：評価値

- ○勝ち＝1.0
- ×勝ち=-1.0
- 引き分け = 0.0 とすると、



モンテカルロ法を○×ゲームに適用

- 例：空いている場所にランダムで○/×を置くシミュレータを使った場合



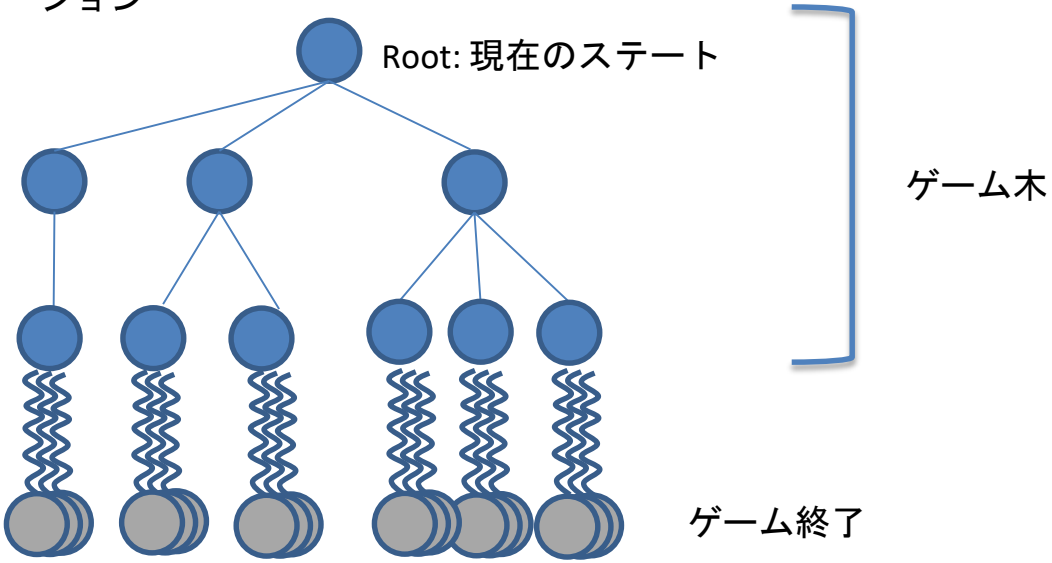
- モンテカルロ法は、全ての状態をノードで表現しきれないゲームに有効

モンテカルロ木探索のDQRへの適用

ゲーム木

ステートノード

シミュレーション

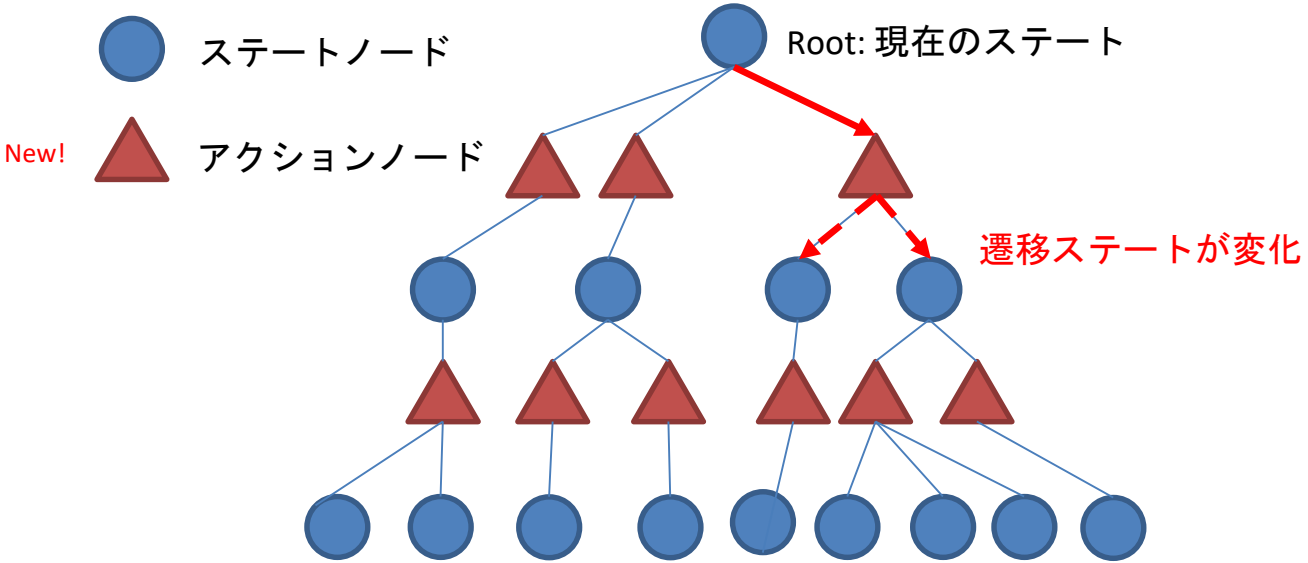


DQR木の課題

不確定・不完全情報をどう扱うか？

DQR木

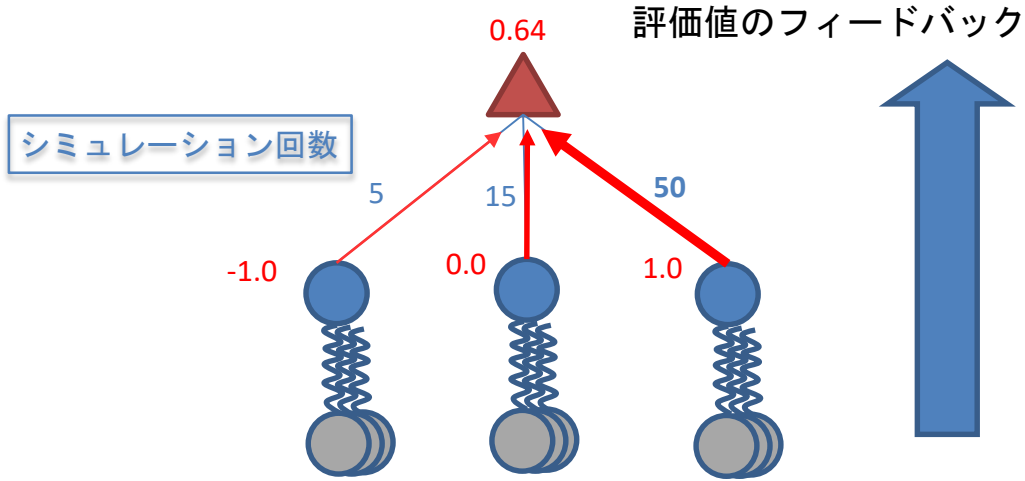
- アクションノードを追加
  - 不確定要素に対応



または
カードドロー等

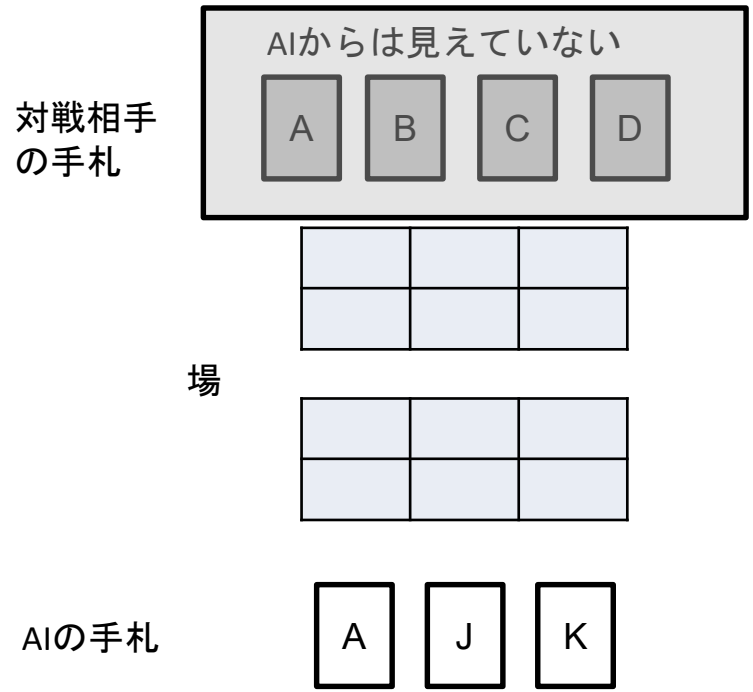
DQR木

- アクションノードの評価値は、子ノードの評価の平均値（期待値）
  - 状態の遷移確率も考慮

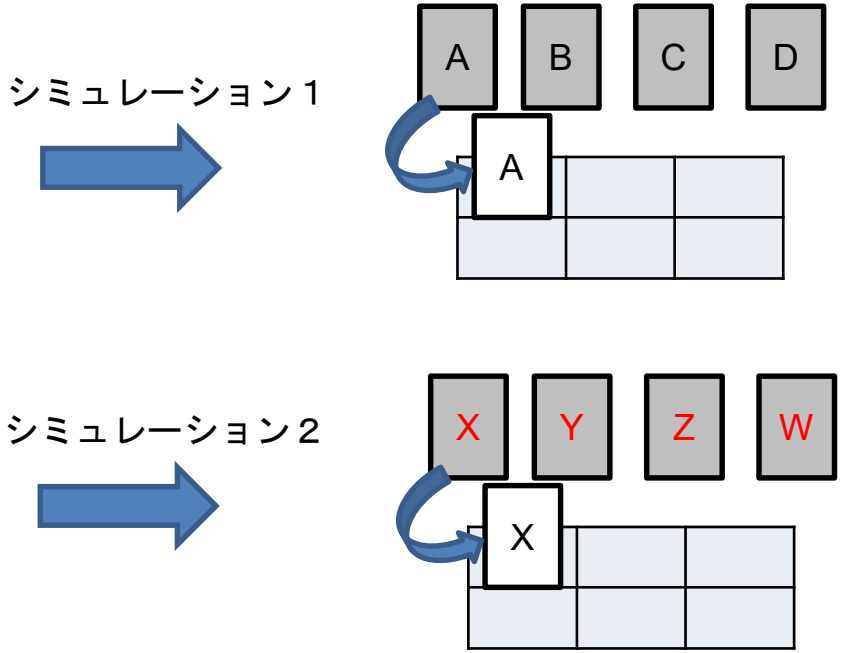


非公開情報のシャッフル

- 山札を含めてシャッフルした後に、シミュレーションを行う

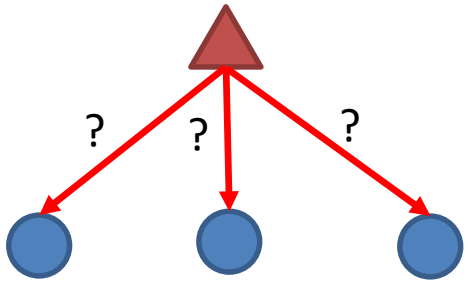


いつも同じカードしかシミュレーションの中で使われないと、AIが手札の内容を間接的に知ってしまう（チート！）



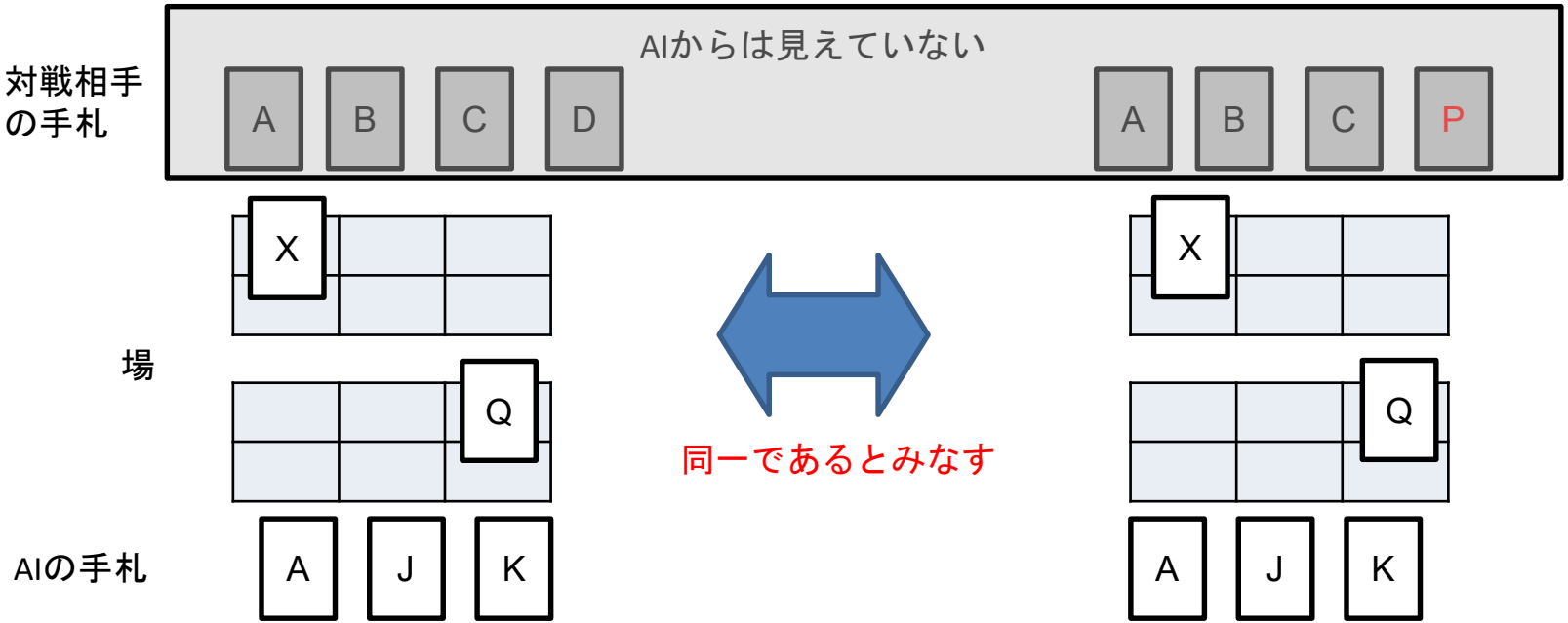
状態の比較

- 行動実行後、既に出現したことがある状態かどうか、調べる必要がある
  - シミュレーション結果の集計のため



同一状態の定義

- AIから見える情報のみを比較する

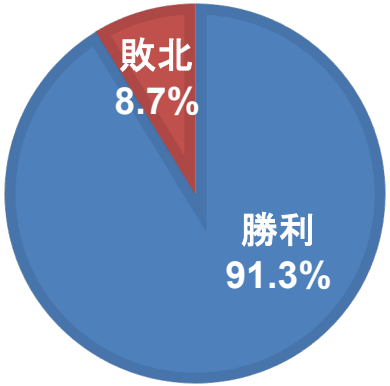


シミュレータ

- シミュレータを別途作成
  - 特殊な操作が必要なため
    - シャッフル
    - 状態の比較
    - 取りうる行動の一覧取得　等
  - 高速化が必要なため
- バトルロジックは、製品版と同一のものを使用

対戦結果

MCTSプレイヤーの
RANDOMプレイヤーとの対戦勝率



* 95%信頼区間: ±0.9%
* MCTSのシミュレーション数は25回/行動

**⇒少ないシミュレーション数で、
ランダムより強いMCTSプレイヤーの作成に成功**

MCTSプレイヤー

- ランダムより強いプレイヤーの作成に成功
  - 比較対象が弱いため、はっきりとは評価できない
  - MCTSも、シミュレーション数を抑えての対戦
    - 25シミュレーション/行動
- 人間からの評価
  - 「トッププレイヤーには及ばない」
    - 1万シミュレーション/行動

MCTS運用の課題

- リソース
  - 269ms/行動
    - 標準的な開発マシンで
  - プレイヤーの待機時間にも影響
- シミュレータの開発
  - 製品版と常に同期する必要がある
  - MCTSの調整はあまり必要ない

DQN

Deep Q-Network (DQN)

- (D)QN
  - 状態空間・行動空間が大きい問題に対し、（ディープ）ニューラルネットワークを使ってQ値の近似を行う手法
  - 事前に評価器を作成するため、ランタイムリソースをあまり消費しない
  - 多くのゲームで成功
  - 多くのライブラリ
    - 本研究ではtensorflowを使用

Q-Learning

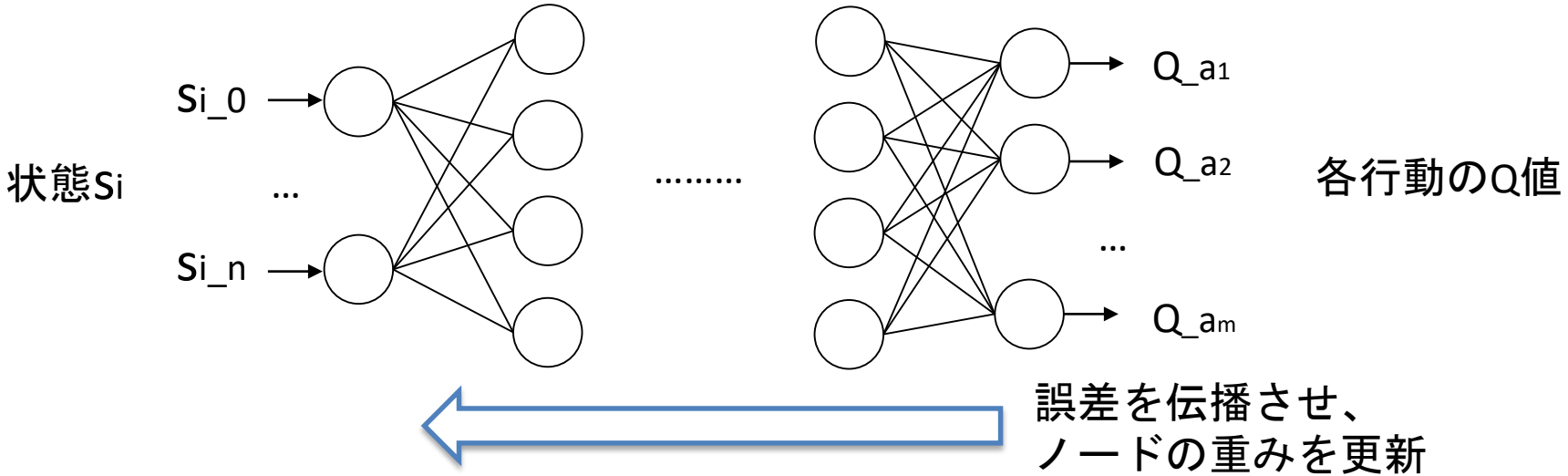
- 全ての状態における各行動についての、Q値を学習する手法の一つ
- Q値: 有効性を表す値

Qテーブル		行動				
		a_1	a_2	a_3	…	a_{10}
状態	S1	0.344	0.213	-0.9223	…	0.208
	S2	-0.620	-0.347	-0.862	…	
	S3	…				
	…					
	S100	0.142	0.345	-0.114	0.446	-0.155

$$Q'(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{古いQ値}} + \underbrace{\alpha}_{\text{学習率}}(\underbrace{r_t}_{\text{報酬}} + \underbrace{\gamma}_{\text{割引率}} \underbrace{\max_a Q(s_{t+1}, a)}_{\text{将来もらえる報酬見積もり}} - \underbrace{Q(s_t, a_t)}_{\text{古いQ値}})$$

DQN

- Q値をネットワークを使って推定



DQNのDQRへの適用

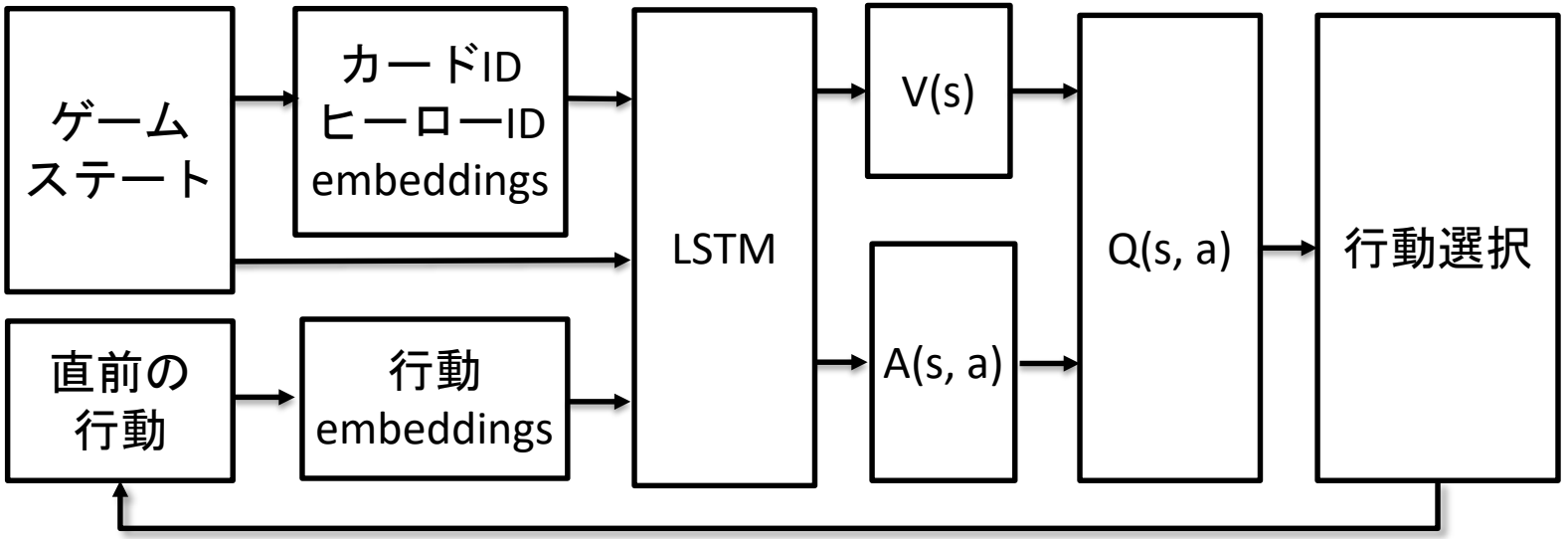
DQNをDQRへ適用するにあたっての課題

- 行動空間が階層的で複雑
  - 複数あるアクションが、さらにそれぞれ様々なパラメータを持つ
- 不完全情報ゲーム
  - 状態を片方のプレイヤーからの視点で記述
  - ニューラルネットワークの不得意分野
- “DQN”と”DQR”が似すぎ
  - 何度も間違えた

拡張

- DQN(λ)
  - より長期的な報酬に焦点を置く
- リカレントネットワーク
  - 直前の行動を入力に追加
    - 不完全情報への対策
    - オフラインデータを利用したADRQNから着想

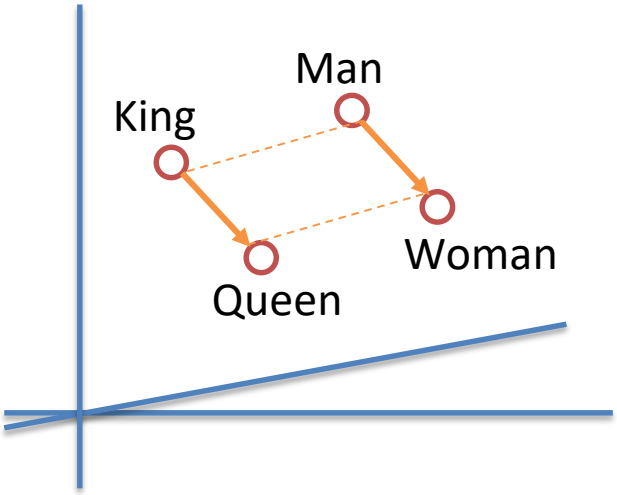
学習構造



- Dueling DQN
- Long Short-Term Memory(LSTM)
- Embedding

Embedding

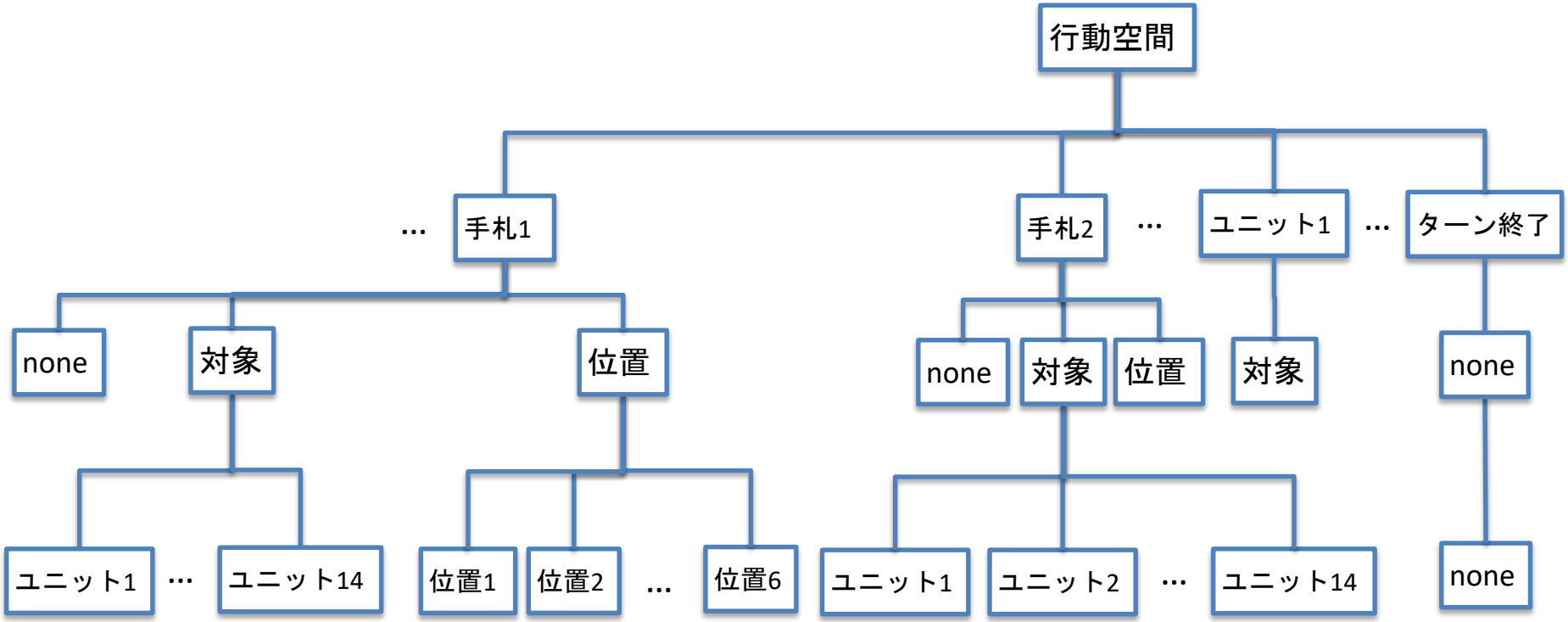
- 特徴をベクトルで表現したもの
  - カードIDなどのカテゴリ入力の、意味的特性を捉えることができる
  - 自然言語処理など、他の分野で一般的



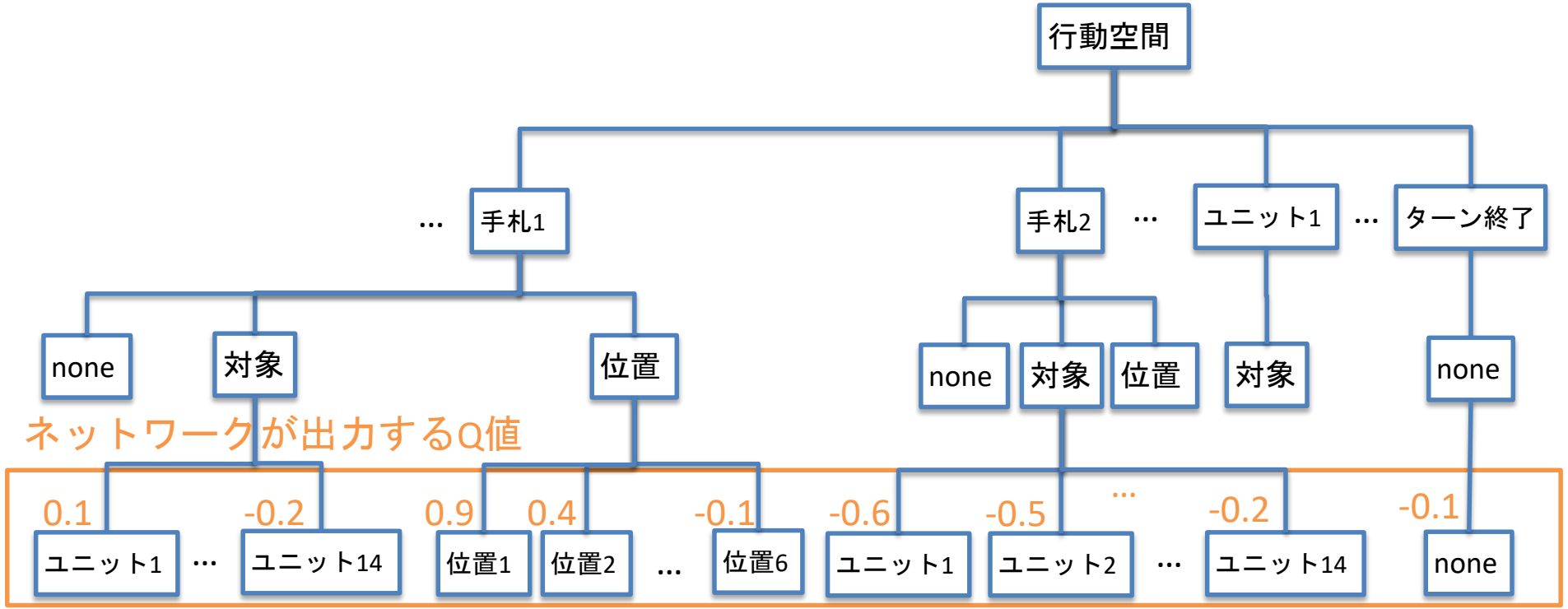
状態空間

- プレイヤーAI
  - キャラクター
  - 武器
  - 手札
  - フィールドユニット
- 対戦相手
  - キャラクター
  - 武器
  - 手札の枚数
  - フィールドユニット

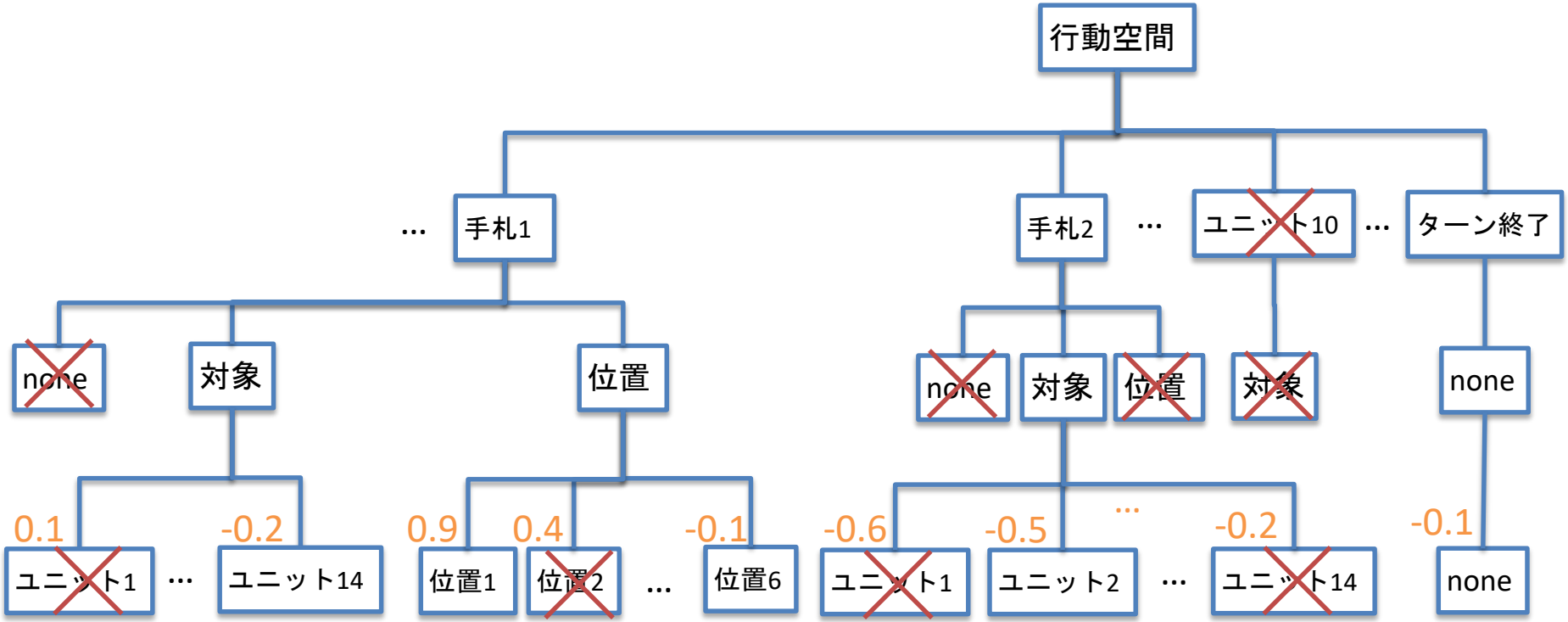
行動空間



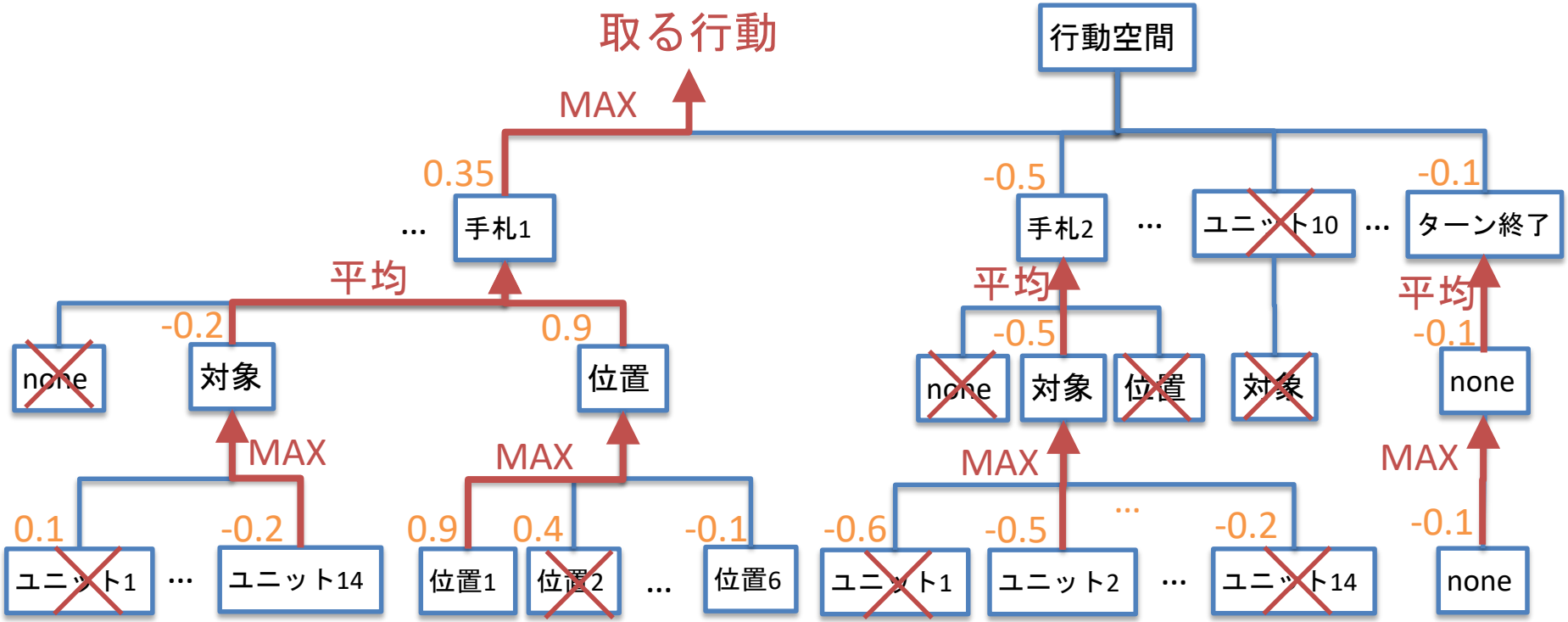
行動空間: ネットワークの出力



行動空間: フィルタリング



行動空間: Q値の決定



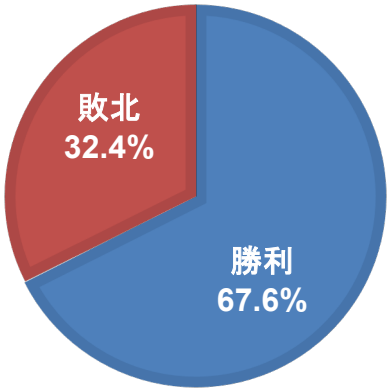
訓練

- シミュレータで生成した50万ログファイルを使用
  - 学習の高速化
  - 将来はユーザーログを利用したい
- バッチサイズ: 200ゲーム
  - 片方のプレイヤー視点から
  - ターン終了で、次に自分のターンが回ってきた状態へ遷移
    - ターン終了＝対戦相手が一連の行動を取る、と同等の意味
- スライスサイズ：20
  - 20の連続した行動の損失を基に、重みを更新

結果

- MCTSプレイヤーに勝利

改善されたDQNプレイヤーの
MCTSプレイヤーとの対戦勝率



95%信頼区間: ±2.9%
シミュレーション数は25回/行動

DQNプレイヤー

- ランタイムリソースを大幅に削減
  - MCTS: 269ms/行動
    - シミュレーション数を増やすと、線形増加
  - DQN: 30ms/行動
    - 半分は、データの変形処理
    - ネットワークが大きくなっても、増加は微小

DQNの課題

- ゲームの更新ごとに、再学習が必要
  - Embeddingsの利用により、効率化は行われている
- 大幅なルール変更があった場合、ネットワークの構造変更が必要

まとめ

まとめ

- MCTSとDQNを使って、DQRをプレイするNPCを作成
  - MCTSプレイヤー
    - より少ないチューニングコスト
  - DQNプレイヤー
    - より少ないランタイムリソース
- 複雑な不完全問題に対し、今後の可能性を提示

Future work

ゲームバランス調整への応用

- キャラクターやカードのバランスを可視化

Randomプレイヤーの
Randomプレイヤー対する先攻時の勝率

	対テリー/ Random, 後攻	対トルネコ/ Random, 後攻
テリー / Random, 先攻	46.3 ± 3.0%	35.0 ± 3.0%
トルネコ / Random, 先攻	63.7 ± 2.9%	52.3 ± 3.0%

MCTSプレイヤーの
MCTSプレイヤーに対する先攻時の対戦

	対テリー/ MCTS, 後攻	対トルネコ/ MCTS, 後攻
テリー / MCTS, 先攻	53.8 ± 3.1%	67.2 ± 2.9%
トルネコ / MCTS, 先攻	48.2 ± 3.1%	61.8 ± 3.0%

シミュレーション数は25回/行動

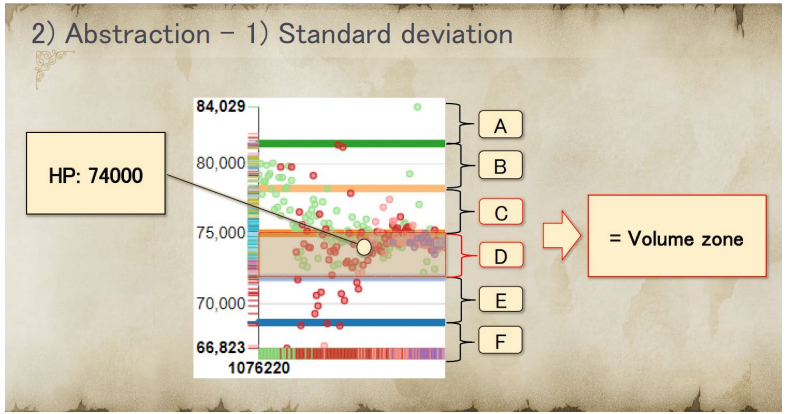
⇒ プレイヤーの習熟度が低い場合、
トルネコの方が強い(※)

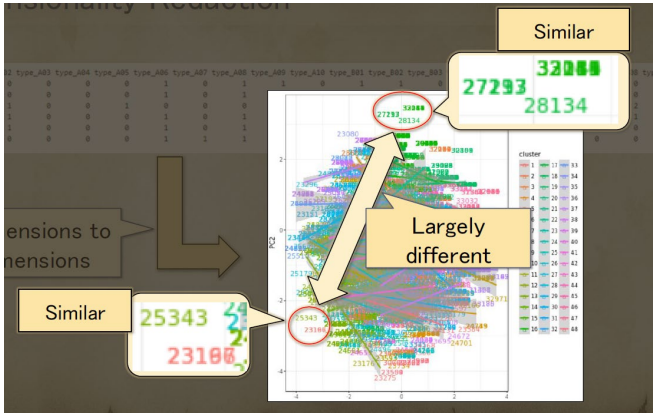
⇒ 習熟度が上がると、トルネコ後攻が弱い
また、テリーの方が強い(※)

(※)限られたカードしか実装されていないシミュレータの場合。
製品版では環境が違うため、同様のことは言えません。

ゲームバランス調整への応用

- 強い戦術をAIに探索させることで、バランスブレイカーを発見
  - 発見された戦術は、デザイナーの意図通りか？
  - ログをデータ解析
  - Kazuko Manabe, “Balancing Nightmares: An AI Approach to Balance Games with Overwhelming Amounts of Data”, GDC2019 (2019)





商標

- TOTAL WARは、ザ クリエイティブ アッセンブリー リミテッドの商標または登録商標です。
- TENSORFLOWは、Google LLCの商標または登録商標です。
- その他掲載されている会社名、商品名は、各社の商標または登録商標です。