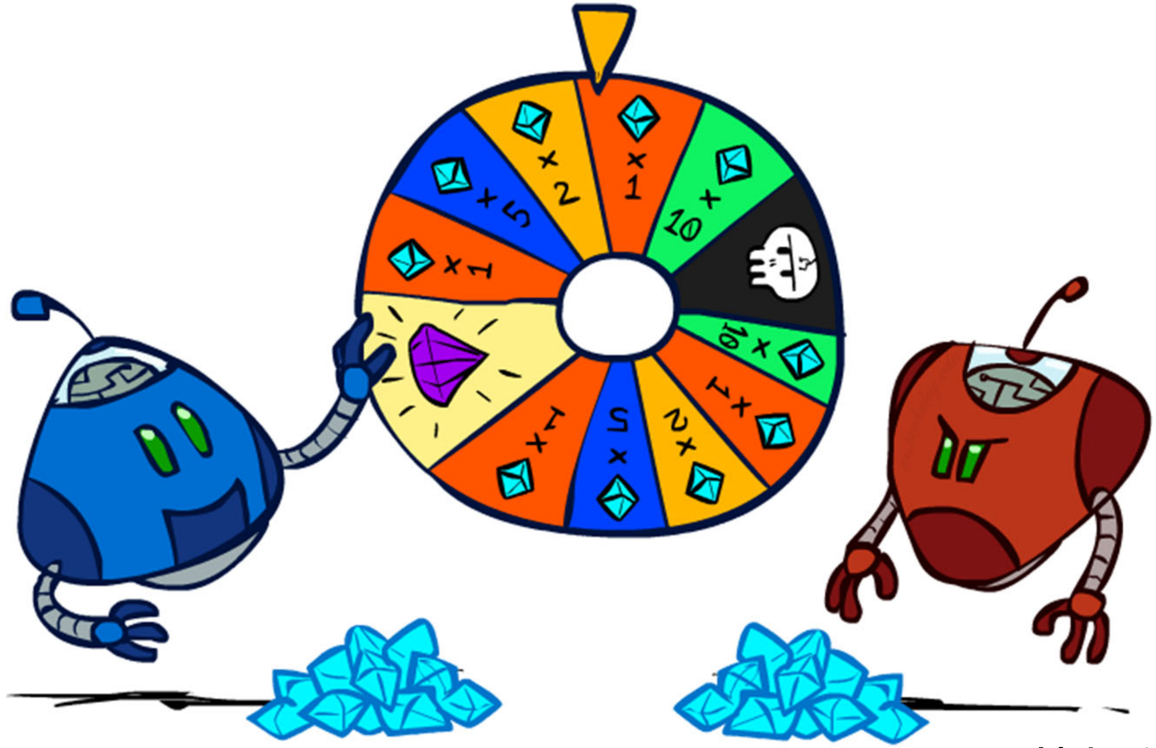
京都大学　情報学科 3回生向け講義「人工知能」

不確かさと効用



情報学研究科　教授　神田崇行
kanda@i.kyoto-u.ac.jp

[Original slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

再掲

議論

- 現状
  - 二人ゲームに勝つ人工知能エージェントができるようになってきた。（二人ゲームを解くアルゴリズムは完成した）
  - 当初の人工知能研究の狙いは、人のような知能を人工的に作る、ということにあった。これは実現できたか？

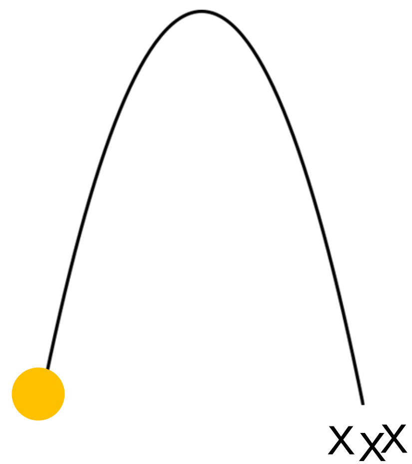
  ⇔人よりも四則演算が速いコンピュータはずっと前からできていた。
  これを「知能」だとは考えてなかった。

- 人工知能は実現できたか？　どう思いますか？

不確かさ (uncertainty)

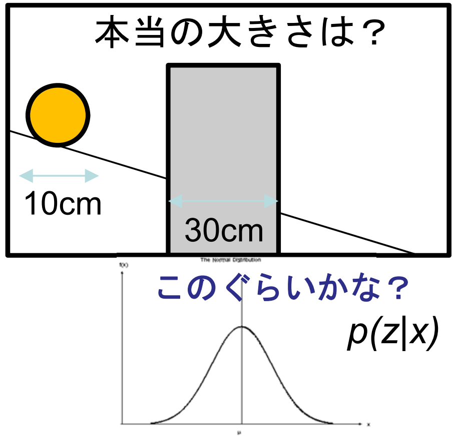
実世界における例

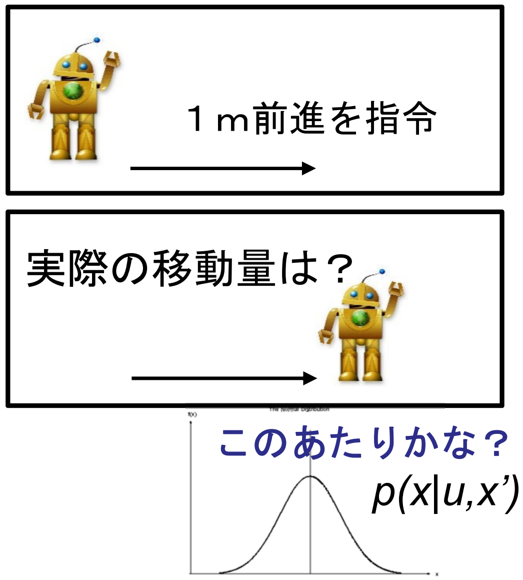
- ボールを投げたらどこに落ちるか？
  - 同じ初速、角度で投げたとしても
  - 環境要因（風など）、制御要因（ちょっとしたズレ）
- 電子メール
  - 例えば、送られてくる文章の内容から、スパムメールを判別したい
  - いつも同じ文章がとどくか？



不確かさ (uncertainty) の例

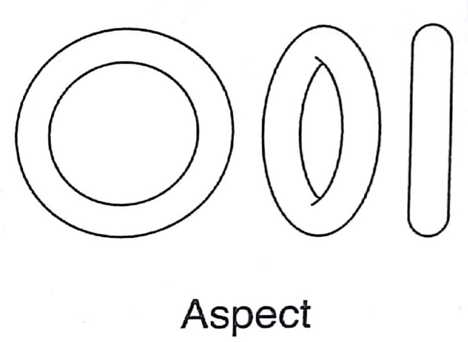
- 観測
  - センサが感知できるものには限界がある
    （有効範囲、分解能、ノイズ）
- 制御
  - モータの予測不可能性
    （制御ノイズ、消耗）
  - 環境とのインタラクション

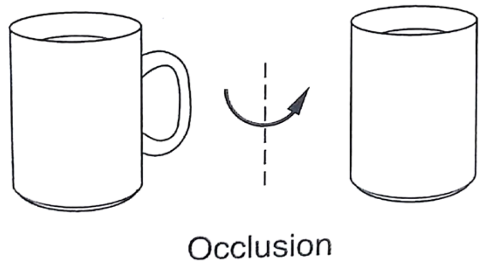




視覚情報処理における不確かさの例

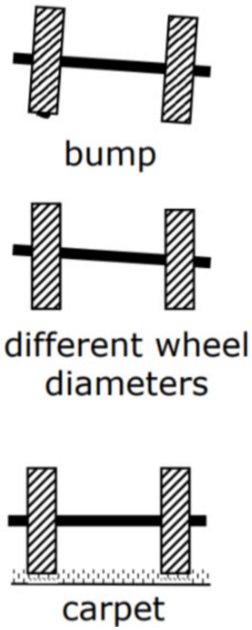
- 見え方 (aspect)
- 遮蔽（occlusion）
- 照明条件
- 個体差
- 背景

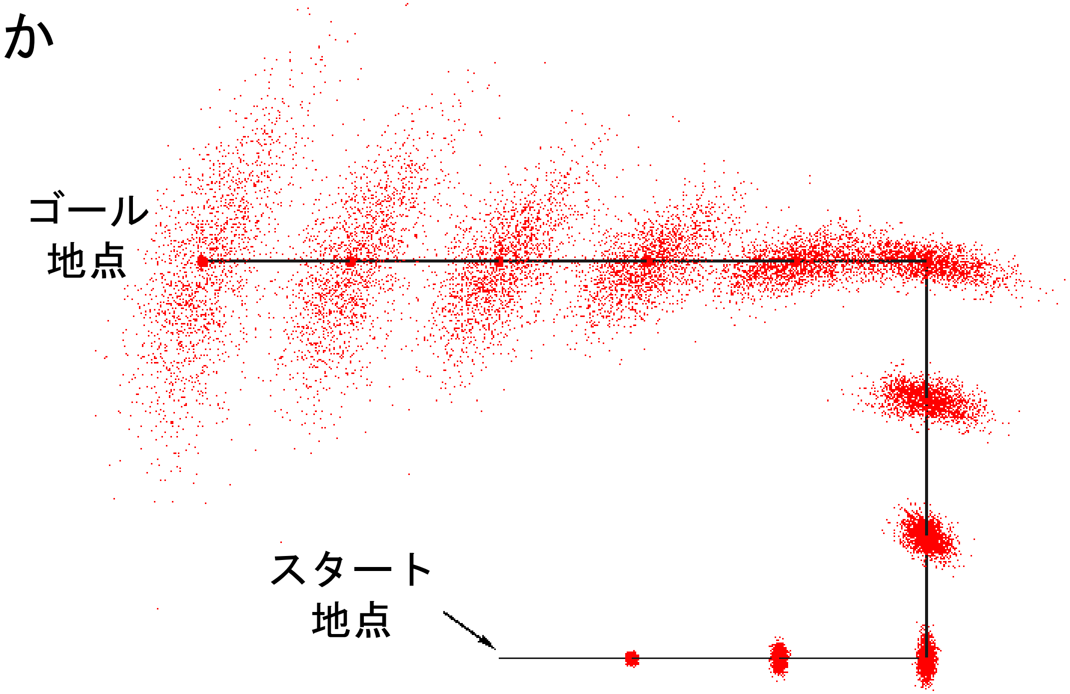




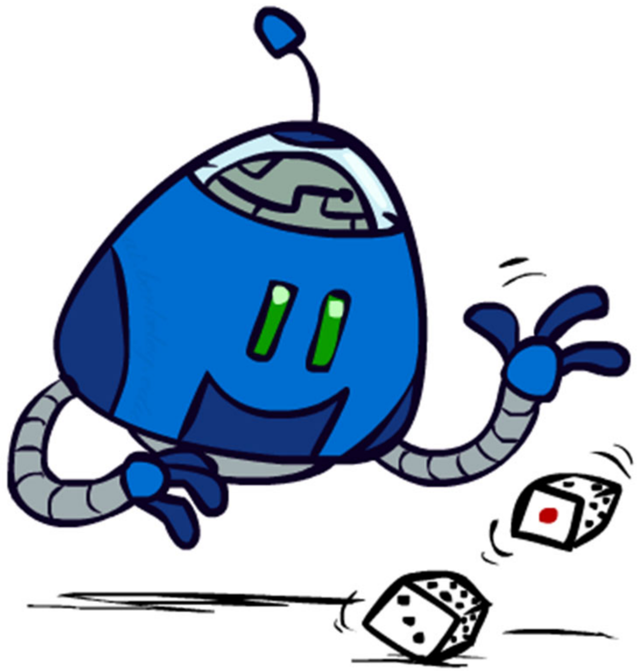
アクチュエーションにおける不確かさの例

- 下図の「スタート地点」から「ゴール地点」まで行きたい
- 「３回直進、左折、３回直進、左折、５回直進」→たどり着くか

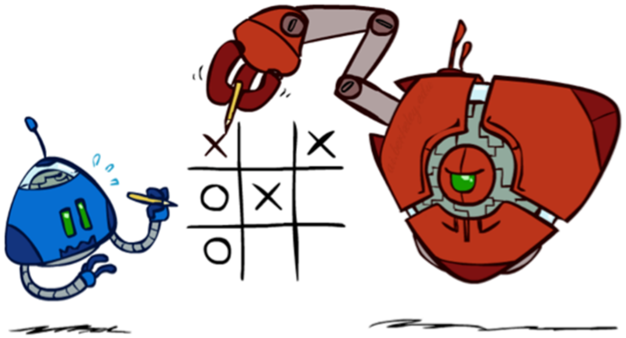


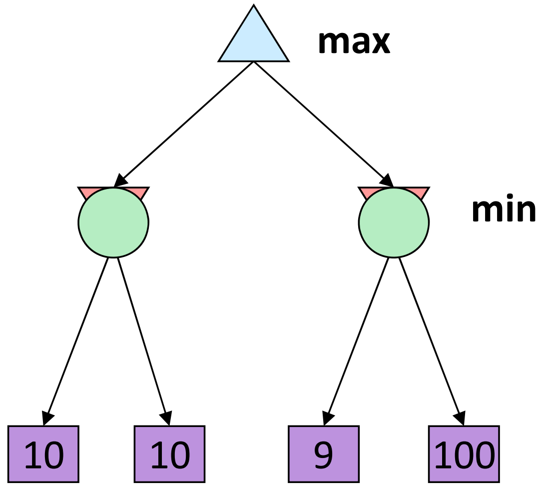


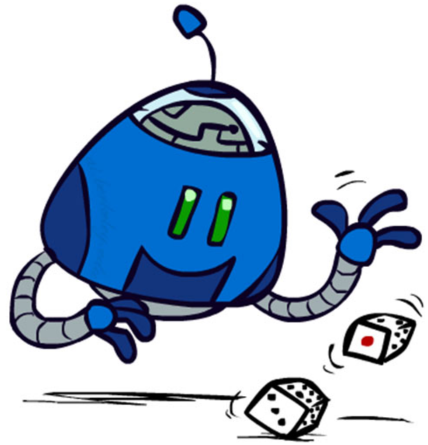
不確かな結果



ワーストケース vs. 平均的なケース



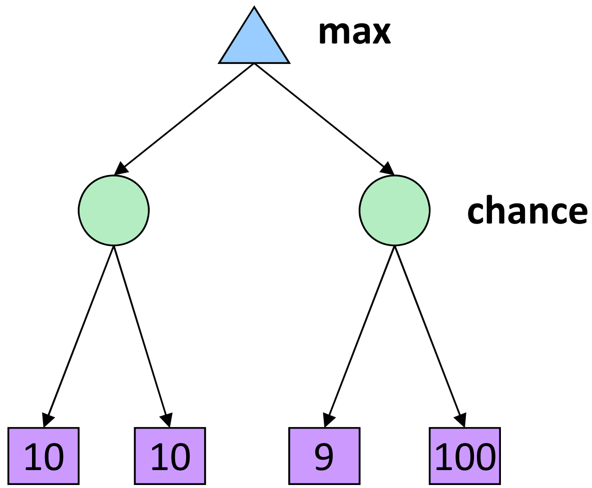




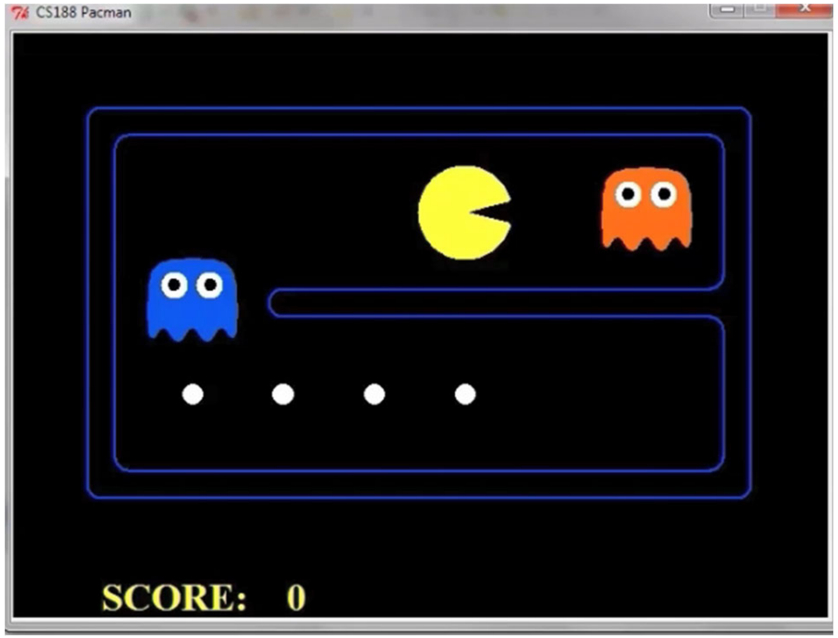
考え方: 不確かな結果は、敵対的な行動ではなく、偶然によって起きる!

期待マックス探索 (Expectimax search)

- 行動の結果を知らないのはどういう場合か？
  - 明示的な偶然性: さいころを振る
  - 相手の予期できない行動: 偶然性で動く敵　（ゲーム等）
  - 行動が失敗するかも: ロボットの移動時に時々スリップ
- 価値（評価値）は、最悪の場合（minimax法）ではなく、平均的な結果を反映すべき (expectimax法)
- 期待マックス探索: 最適な行動での平均的な利益を計算
  - マックス節点はミニマックス探索と同じ
  - チャンス節点は、結果が「不確か」な節点
  - ここでは、期待効用を計算
  - つまり、子節点に予期する結果の重み付き平均
- 後の講義では、マルコフ決定過程として、予期しない結果が起きるような問題における意思決定を扱います

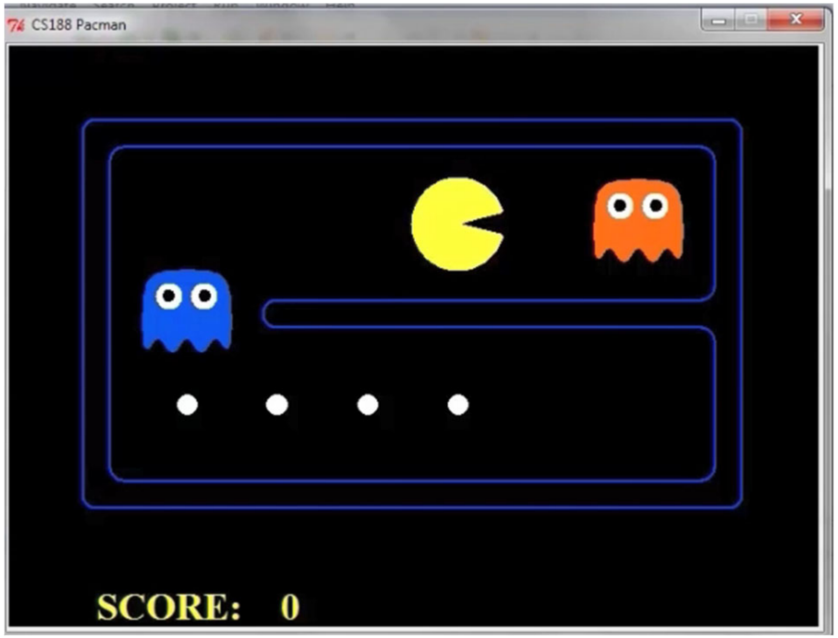


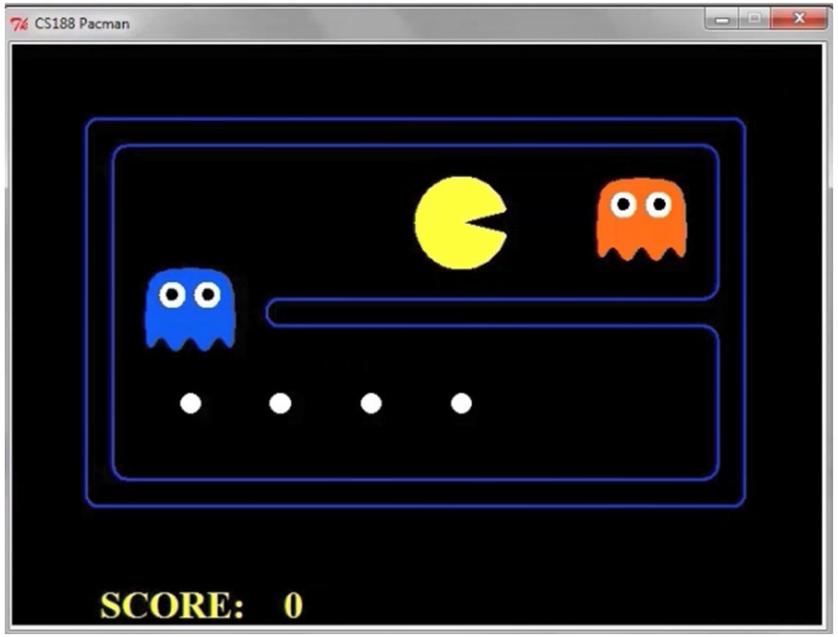
ミニマックスと期待マックスの違い



ミニマックス法

ミニマックスと期待マックスの違い





期待マックス法

期待マックス法　疑似コード

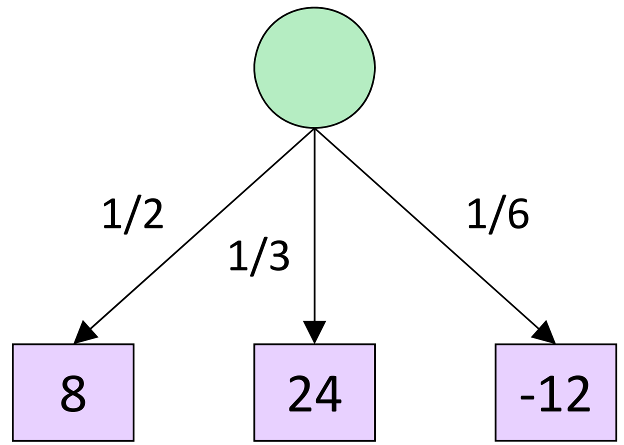
```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

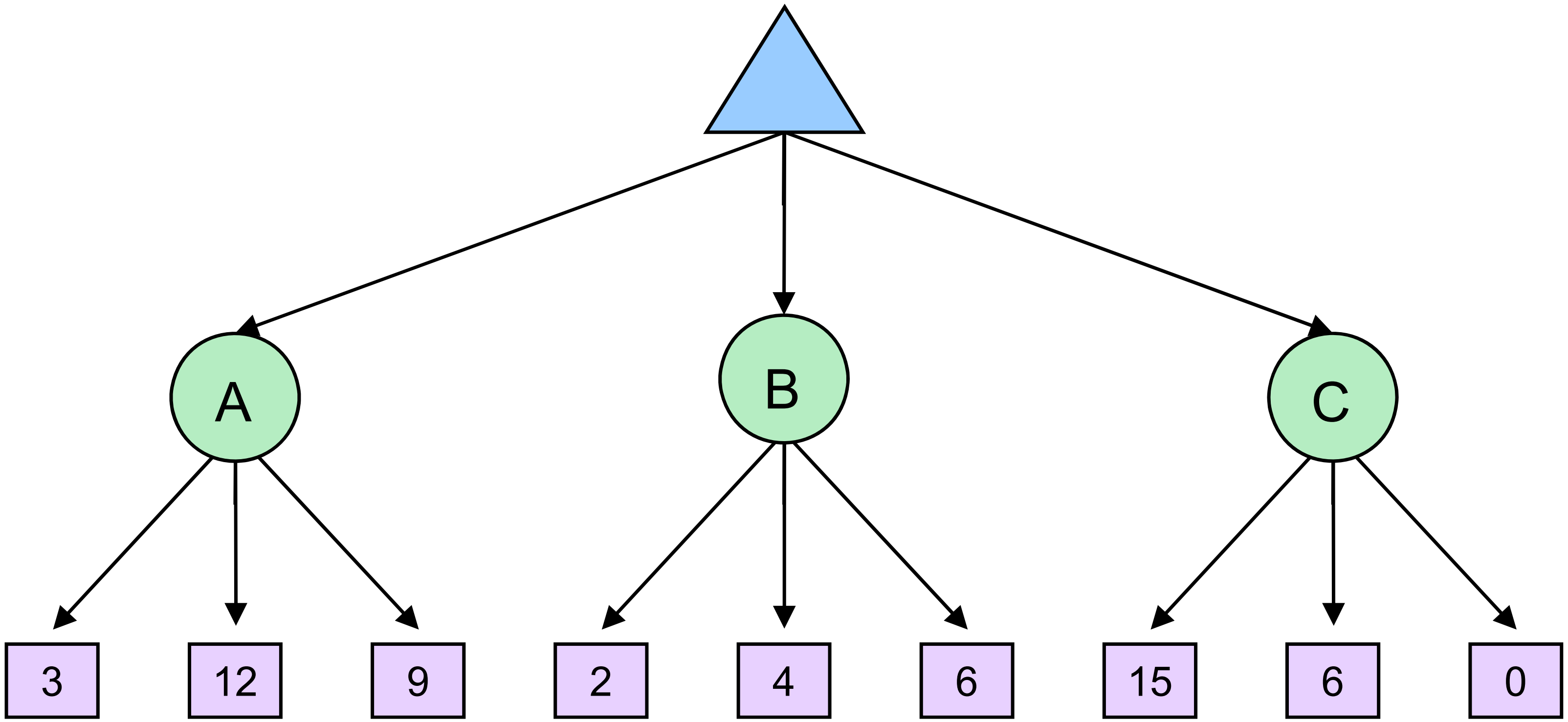
期待マックス法　疑似コード

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

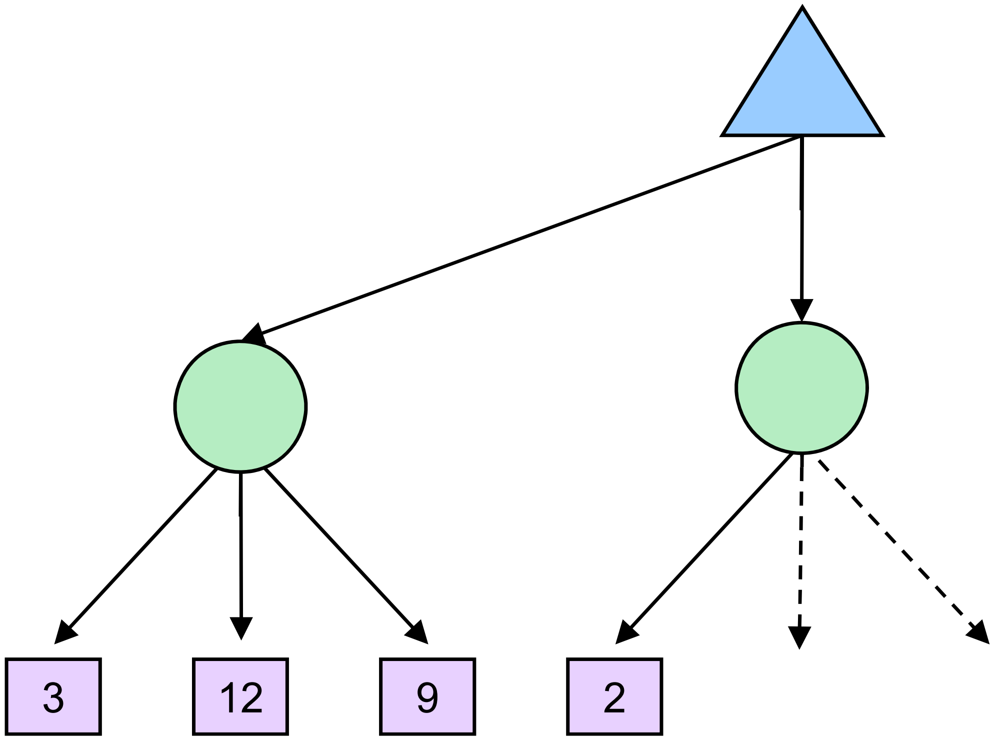


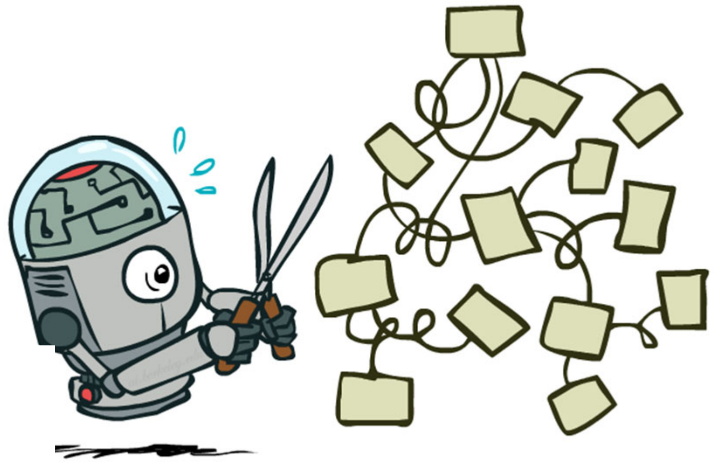
v = (1/2) (8) + (1/3) (24) + (1/6) (-12) = 10

例題：期待マックス法

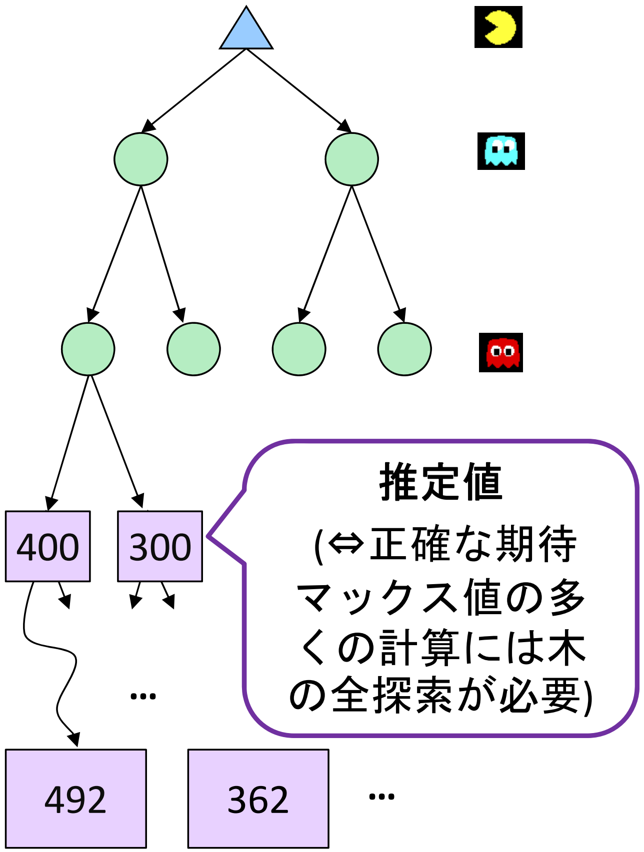


期待マックス法　枝刈りは可能？

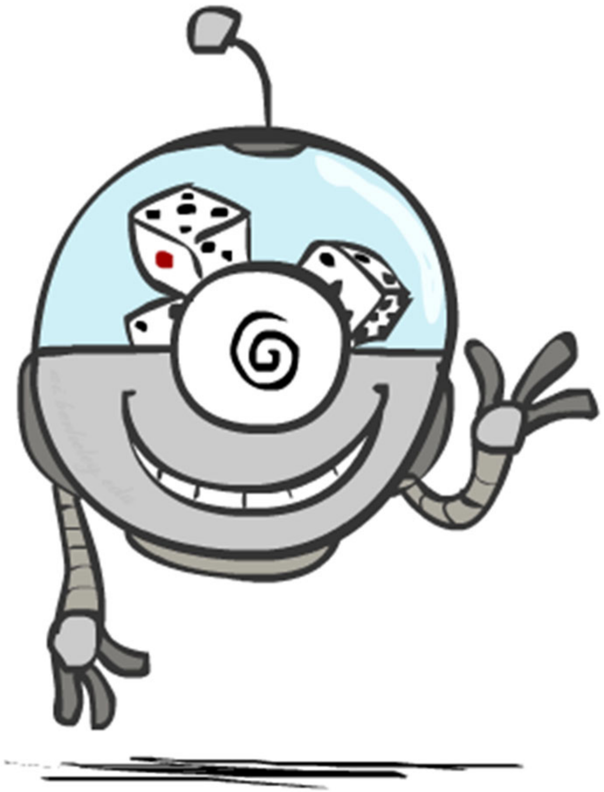




深さを限った期待マックス法

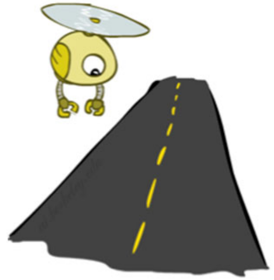


確率



復習：確率(probability)とは？

- 確率変数 (random variable) は、結果が分からないイベントを表す
- 確率分布（probability distribution）結果への重みづけ
- 例: 道路の交通量
  - 確率変数: T = 通行があるかどうか
  - 結果: T in {none, light, heavy}
  - 分布: P(T=none) = 0.25, P(T=light) = 0.50, P(T=heavy) = 0.25
- 確率に関する法則 (もっと詳しくは後程):
  - 確率は負にならない
  - すべての取りうる結果についての確率を合計したら1になる
- より多くの証拠(evidence)が分かると, 確率は変わりうる:
  - P(T=heavy) = 0.25, P(T=heavy | Hour=8am) = 0.60
  - 確率に関する推論や、確率の更新に関する方法は後程習います



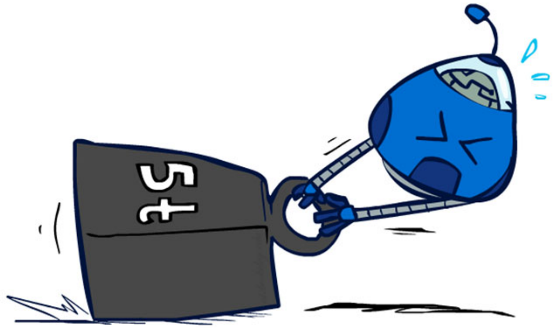
0.25

0.50

0.25

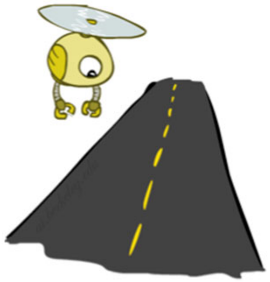
復習: 期待値とは？

- 期待値とは、確率変数を確率分布により重みづけ平均したもの



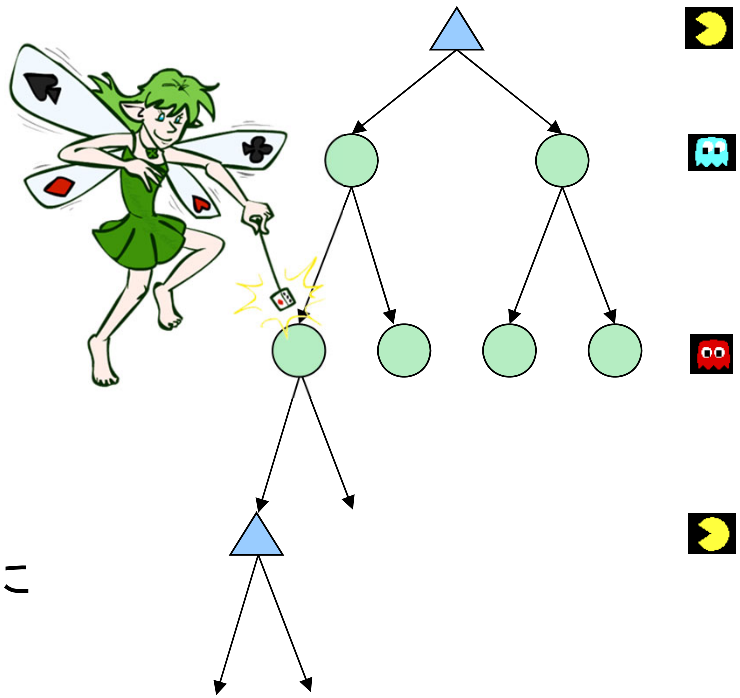
- 例: 空港まで行くのにかかる時間

時間:	20 min		30 min		60 min		
	x	+	x	+	x	⇒	35 min
確率:	0.25		0.50		0.25		



どの確率モデルを使うべきか？

- 期待マックス探索において、相手（あるいは環境）が各状態でどのようなふるまいを見せるかの確率モデルは？
  - 単純な均一分布モデル（さいころを振る）
  - より丁寧に構築されたモデル（多くの計算が必要）
  - 自分たちでは制御できないチャンス節点：
    相手の動きによるもの、環境要因
  - モデルは、敵対的な行動がどのようになりそうか、についての情報をもたらす！
- 今のところは、チャンス節点は何らかの方法で、結果に関する確率分布をもたらすものである、として先に進んでいきましょう

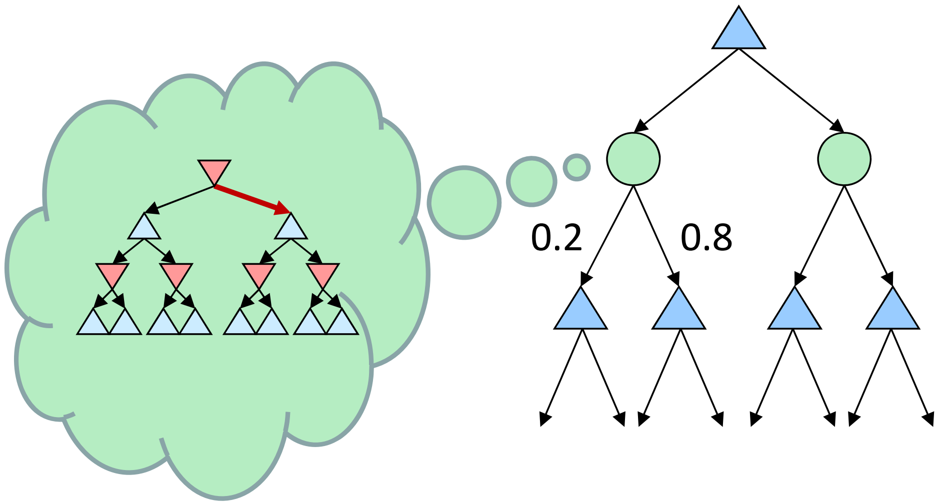


他のエージェントを確率的に行動するというモデルで扱う

≠ 本当にそのエージェントがランダムに行動している!

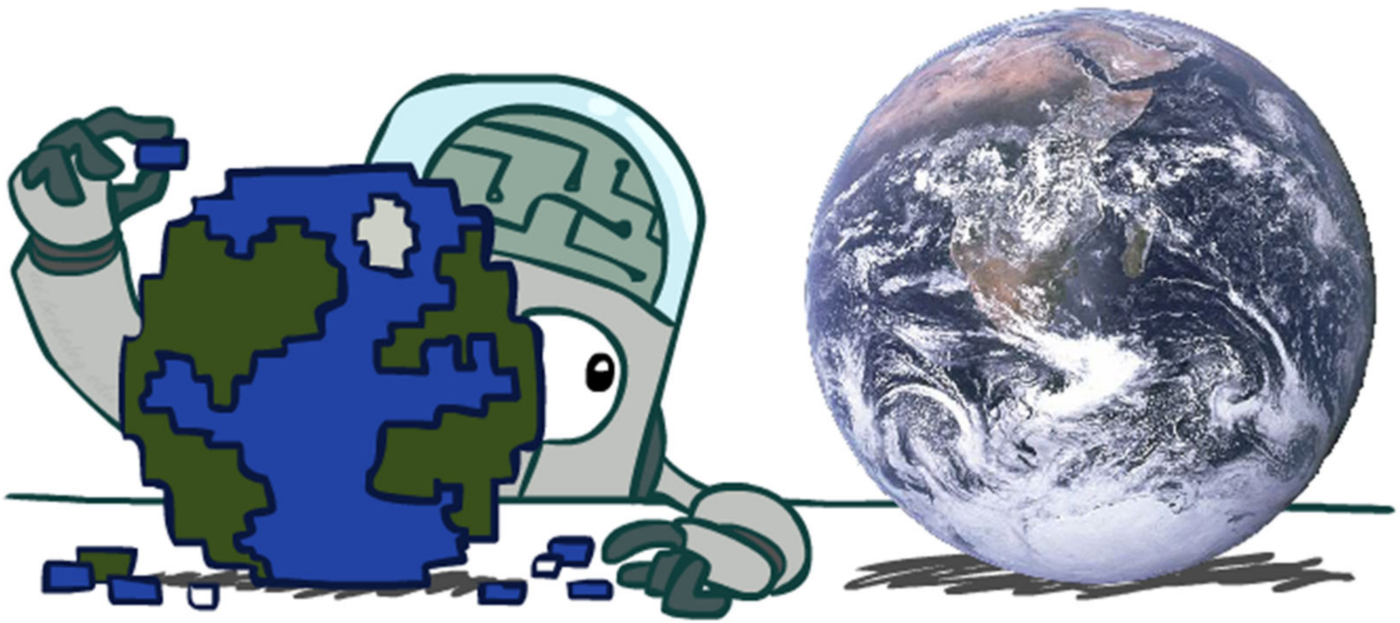
こんな場合は？ Informed Probabilities

- 仮に、相手のエージェントが、実際に深さ２のミニマックス探索を80%の可能性で行い、20%はランダムに行動しているとしましょう
- 質問: どのような木探索アルゴリズムを用いるべきでしょうか？



- 答え: 期待マックス法!
  - 各チャンス節点の確率を見出すためには、相手のエージェントの行動をシミュレーションする必要がある
  - たいてい、こういった処理には時間がかかる
  - さらに難しいのは、相手のエージェントがあなたの行動をシミュレーションしていることをシミュレーションする必要がある・・・
  - ・・・ただし、ミニマックス法を使えば、こういった特性をゲーム木で表現できる

モデル化における仮定



楽観的/悲観的な見積もりのリスク

楽観的であるリスク

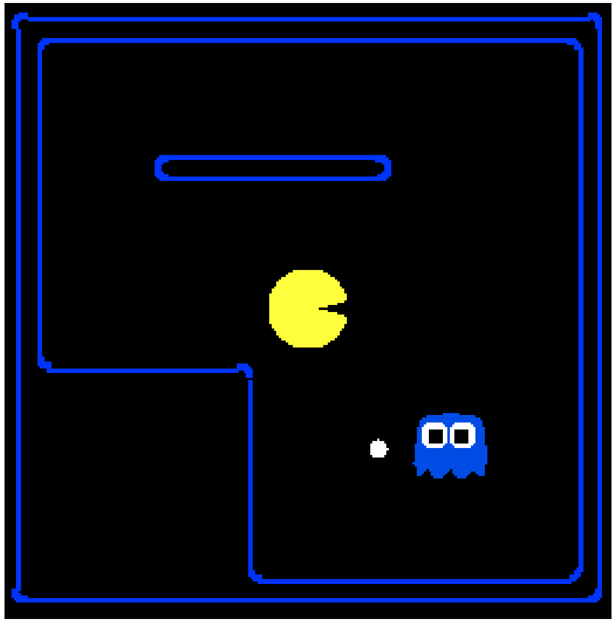
本当は敵対的なのに、偶然だと思ってしまう

悲観的であるリスク

そんなことは無いのに、ワーストケースを考える

実際の現実よりも楽観的/悲観的であると、どのような結果になってしまうでしょうか？

例：仮定と現実の違いの影響

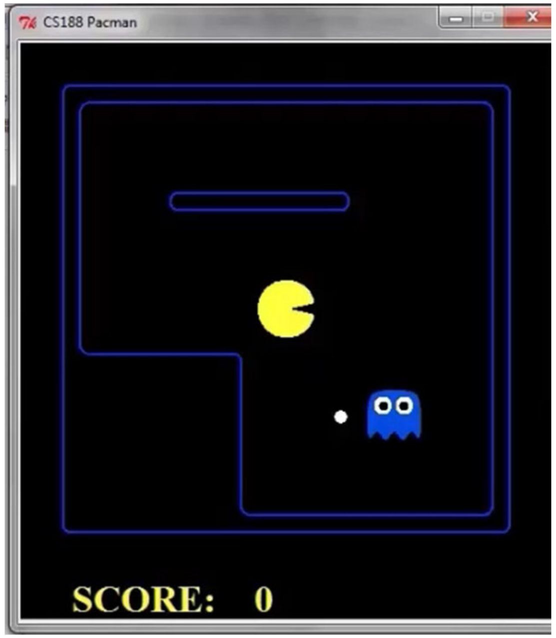


	敵は敵対的に行動	ランダム行動
味方の行動 ミニマックス法	Won 5/5 Avg. Score: 483	Won 5/5 Avg. Score: 493
期待マックス法	Won 1/5 Avg. Score: -303	Won 5/5 Avg. Score: 503

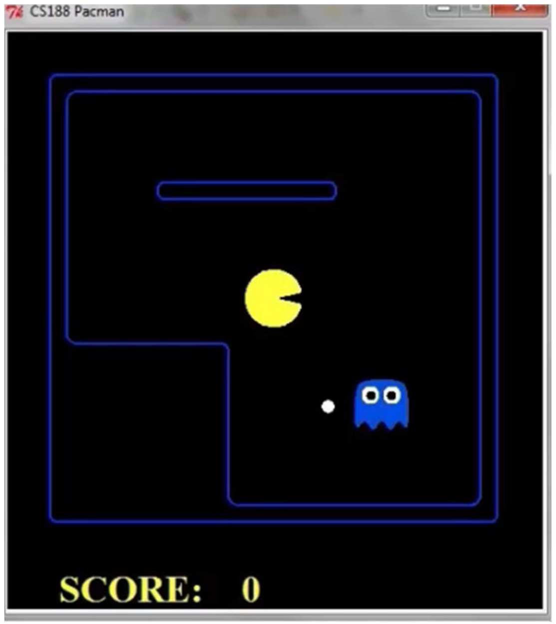
Results from playing 5 games

パックマンは「トラブルを避ける」評価関数を用いて深さ４の探索
ゴーストは「パックマンを見つける」評価関数を用いて深さ２の探索

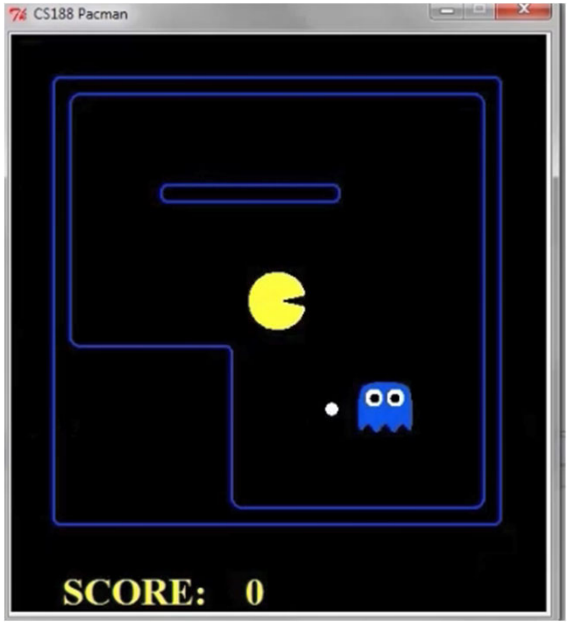
敵：ランダム　味方：期待マックス法



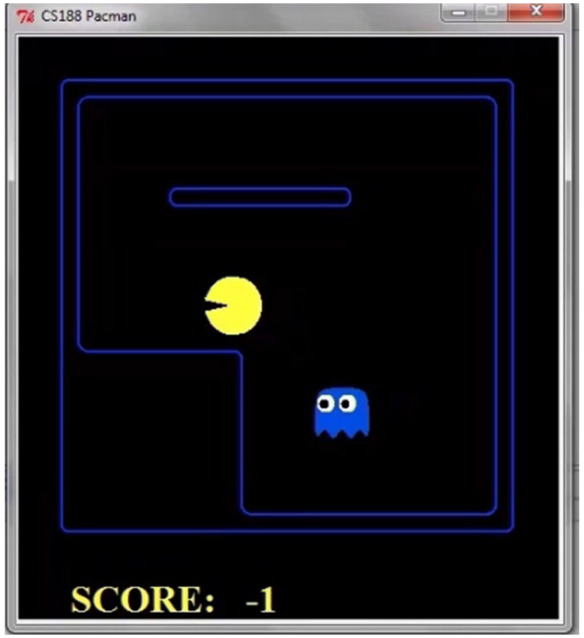
敵：敵対的行動　味方：ミニマックス法



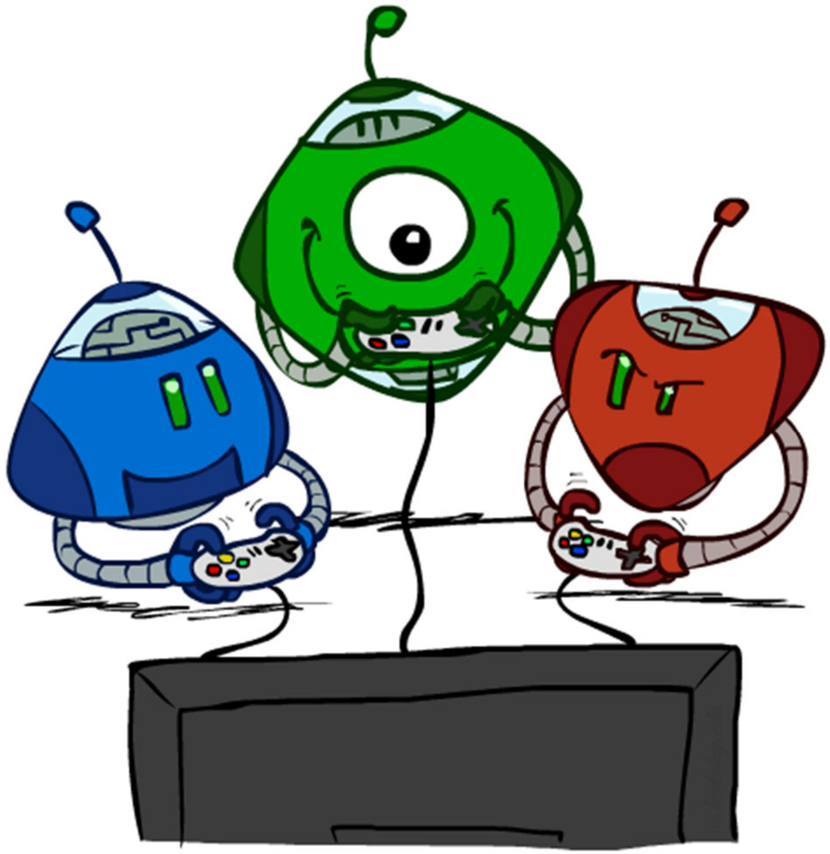
敵：敵対的行動　味方：期待マックス法



敵：ランダム　味方：ミニマックス法

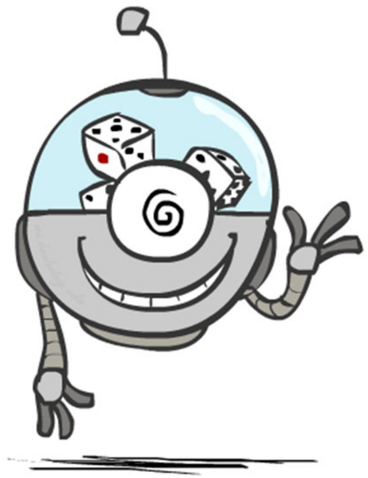


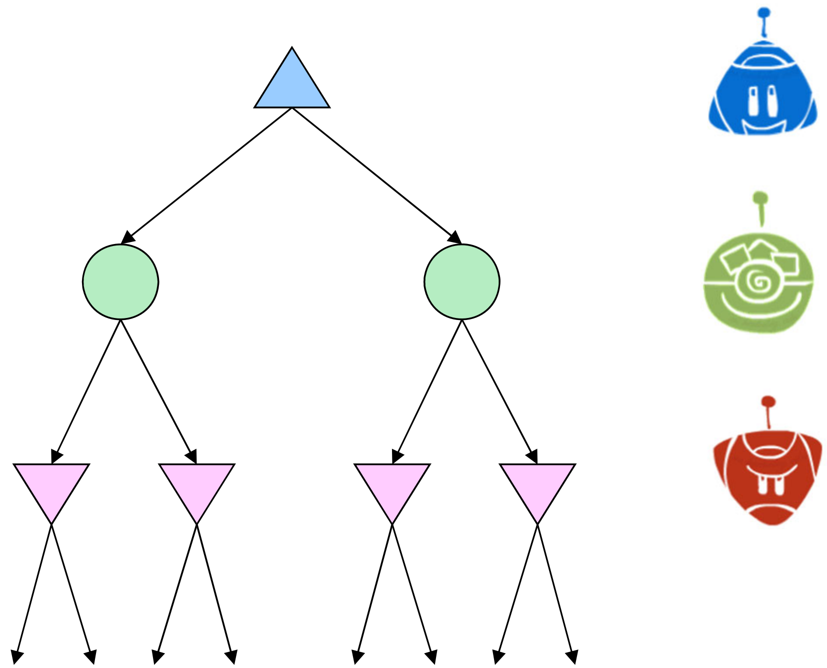
他のゲーム



異なるタイプの層が混じったもの

- 例：バックギャモン
- 期待ミニマックス
  - 環境要因は各プレイヤーの行動後に起きるランダムエージェントとしてモデル化
  - 各節点でそのタイプに応じた処理をして、その子節点の情報を統合



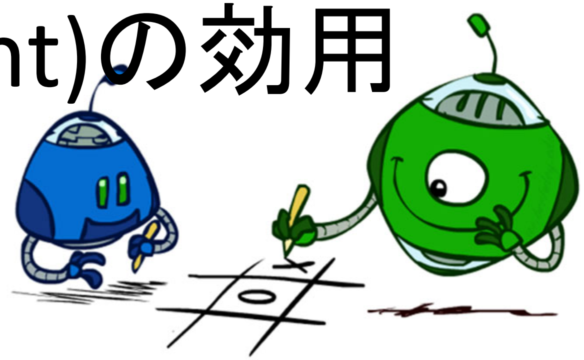


例: バックギャモン

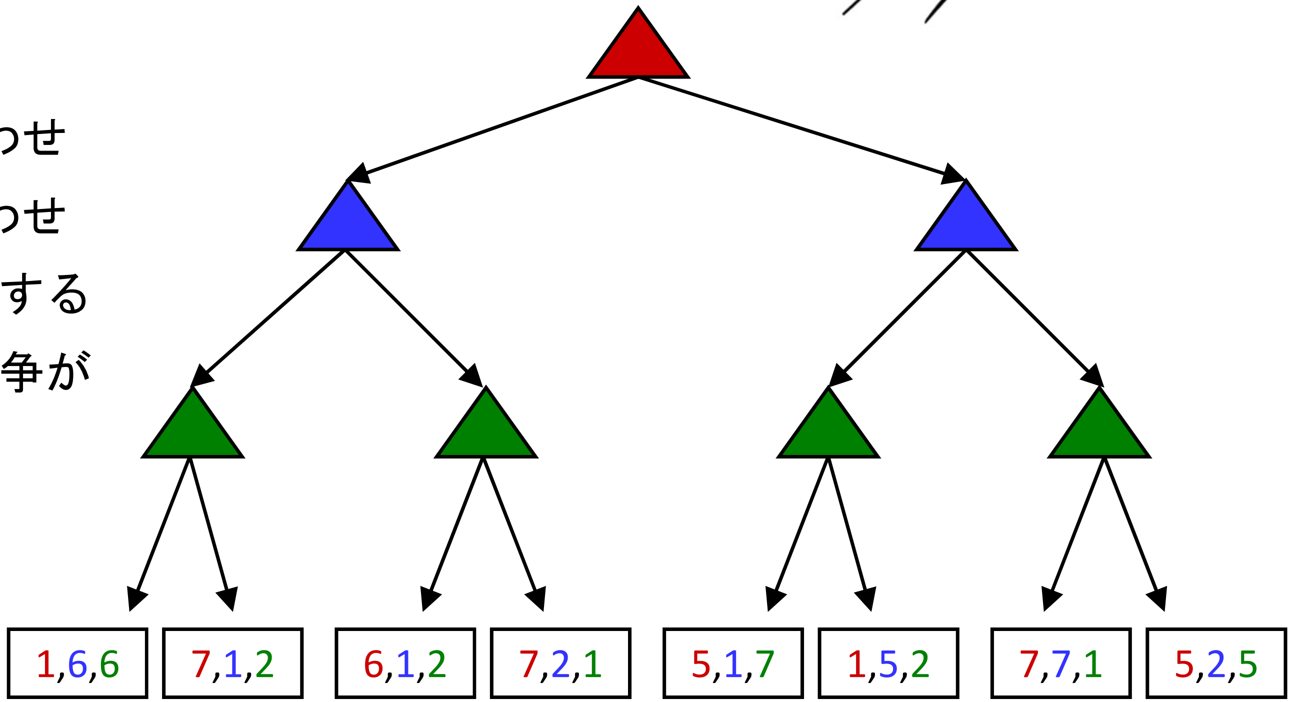
- ２つのさいころを振って移動先を決める
  →分岐数 b: 21 (ぞろ目6種、ぞろ目以外15種)
  - バックギャモン ≈ 20の可能な動き
  - 深さ2の場合 = 20 x (21 x 20)^3 = 1.2 x 10^9
- 深さが増えるにつれて, ある探索した節点に実際に到達する可能性は小さくなる
  - 探索の有効性が減っていく
  - 深さを限ってもそれほど困らないはず
  - しかし、枝刈りは難しい
- 人工知能における歴史: TDGammon は深さがわずか2の探索 + すごく優れた評価関数 + 強化学習：世界チャンピオンに匹敵する実力
- AIがなんらかのゲームでチャンピオンになったのはこれが初めてだった

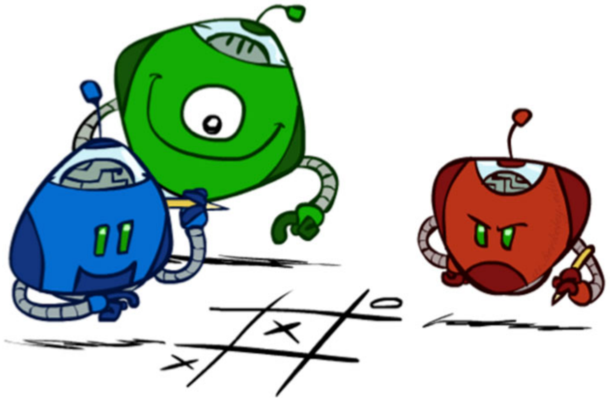
Image: Wikipedia

マルチエージェント(Multi-Agent)の効用

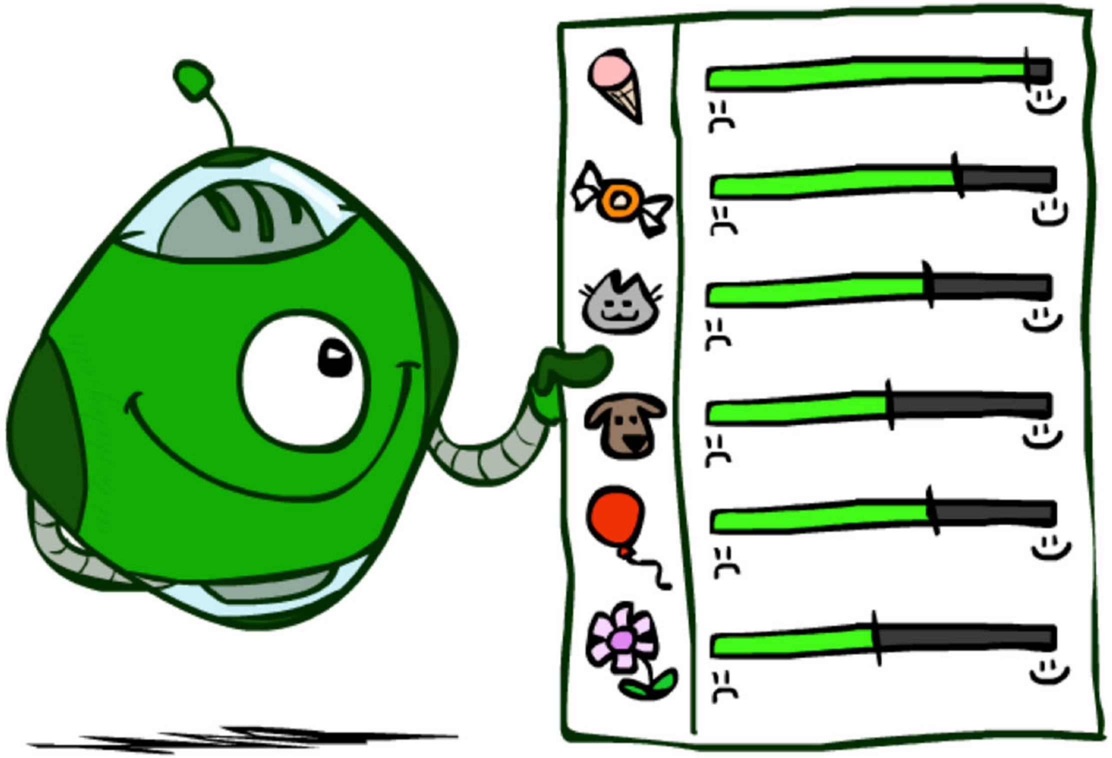


- もしゲームがゼロサム問題ではなく、あるいは複数のプレイヤーがいたら？
- ミニマックスの一般化:
  - 終端節点は3者以上の効用の組み合わせ
  - 中間節点も3者以上の効用の組み合わせ
  - 各プレイヤーは自身の効用を最大化する
  - 結果として、協力が起きるなど、競争がよりダイナミックなものに…



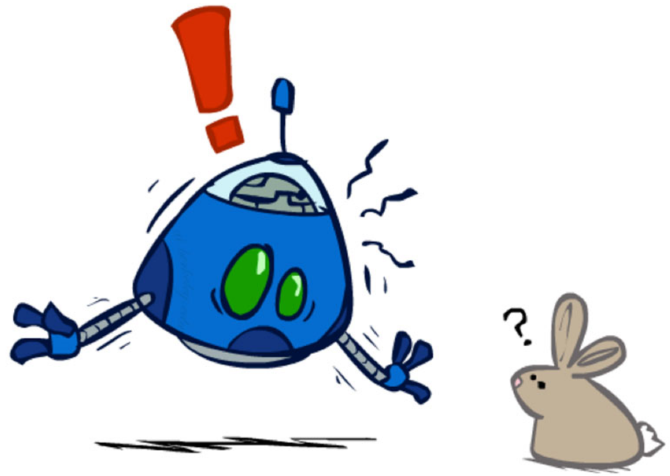


効用 (Utilities)

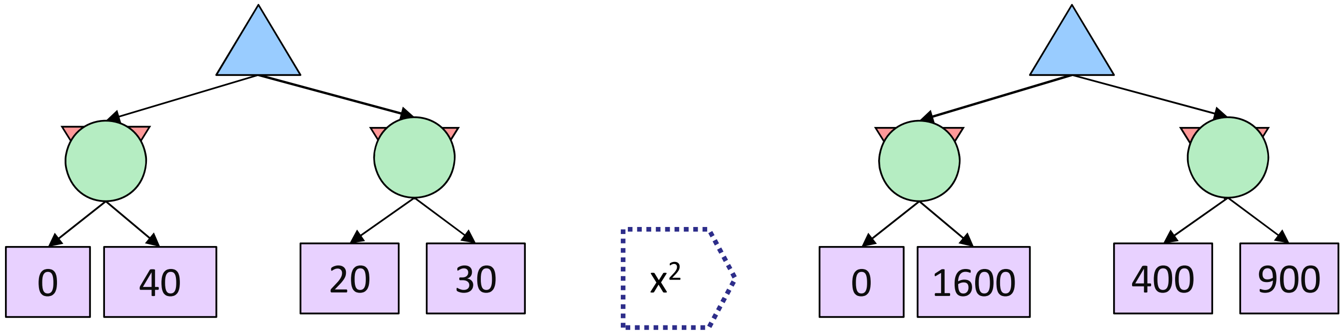


最大の期待効用

- なぜ効用の平均をとるのか? なぜ最大・最小値を考えないのか?　（⇔ミニマックス法）
- 最大期待効用の原則:
  - 合理的なエージェントは、自らの持つ知識に基づいて、期待効用を最大化するように行動する
- 疑問:
  - どうやって「効用」を定義するのか?
  - そもそも「効用」というものは本当に定義できるのか?
  - 「効用」は平均をとれるようなものなのか?
  - 人間の行動（選好）は効用で表現できるのか？



どういった効用（関数）を使うべきか？

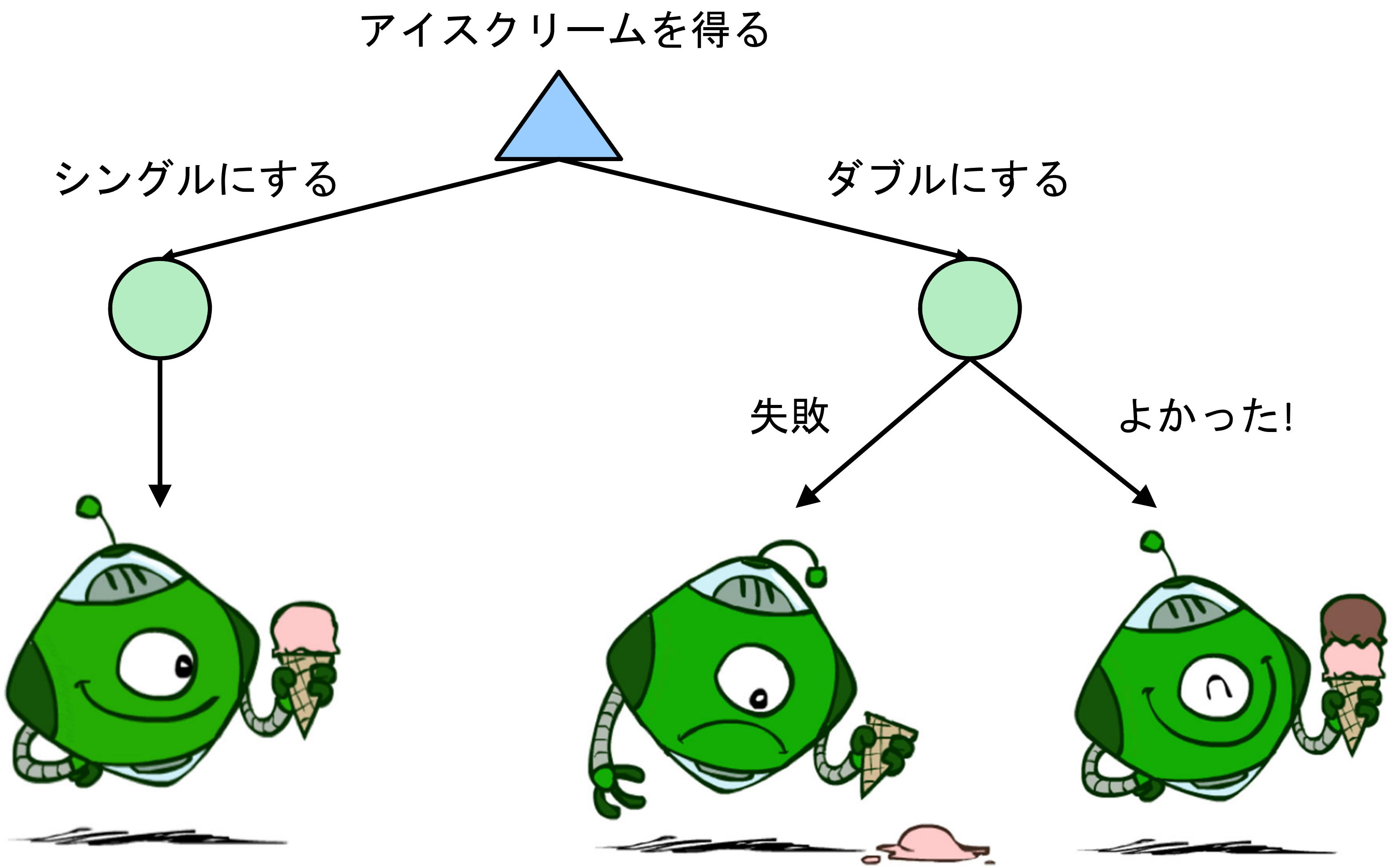


- ワーストケースを考えるミニマックス手続きでは、効用関数の示す値が実際の値に沿っているかどうかは重要ではなかった
  - より高い評価値の状態が知りたい（正しい順序を知りたい）
  - 一様な変形への不感性 (insensitivity to monotonic transformations)があるといえる
- 一方、典型的な期待マックス手続きでは、効用の大きさ（が正しいこと）が重要である

効用とは？

- 「効用」とは、結果（実世界の状態）を、エージェントの選好を表す数字に変換する関数である
- 効用関数をどうやって定義するか？
  - ゲームでは、単純に勝ち負け (+1/-1)
  - エージェントの目標を要約したもの
  - 公理：「合理的」な選好は効用関数として定義できる
- 効用関数を作り、行動を発現させる（入力：効用　→出力：行動）
  - なぜ、ゴールを与えないか？
  - なぜ、行動そのものを記述しないか？

効用: 不確かな結果



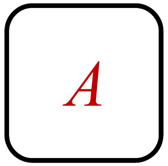
選好 (Preferences)

- エージェントは複数候補の間に選好を持つ：
  - 賞品: A, B, など.
  - くじ引き: どの賞品がもらえるか不確か

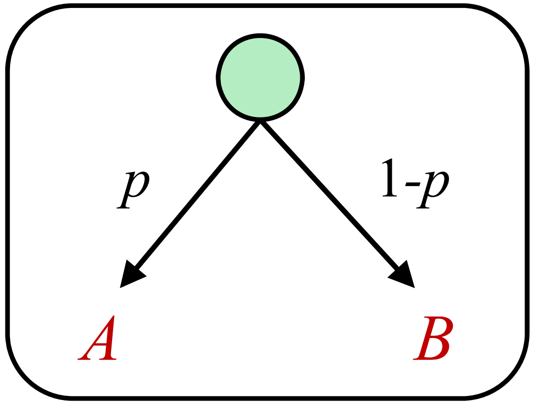
$$L = [p, A;\ (1-p), B]$$

- 表記法:
  - 選好: $A \succ B$
  - どちらでもよい: $A \sim B$

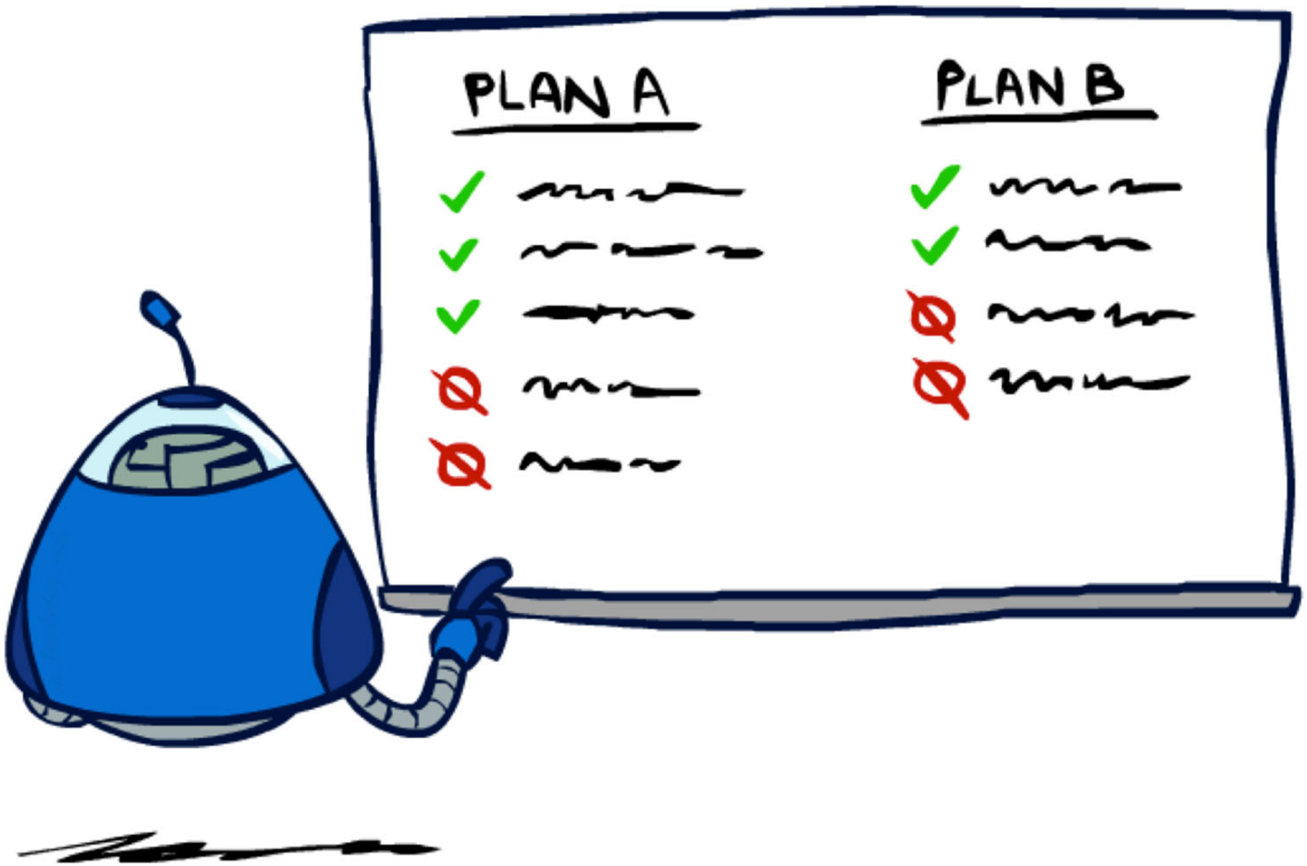
賞品



くじ引き



合理性 (Rationality)

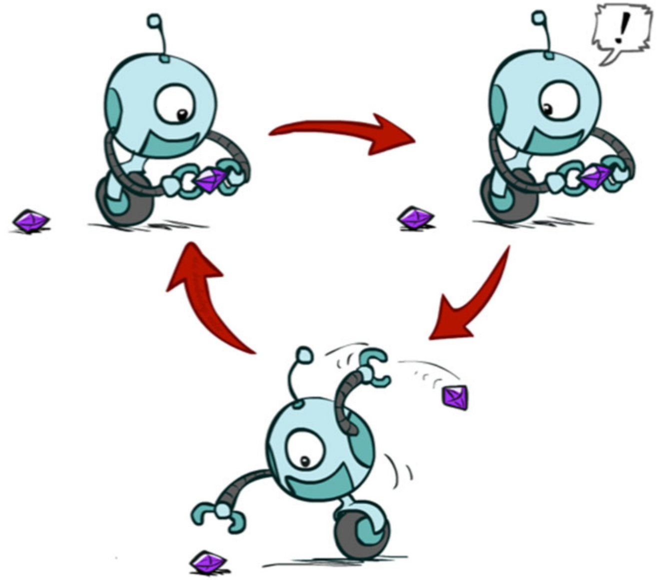


合理的な選好

- 「合理的」であるためには、選好関係に以下のような制約が当てはまるべき：

推移性の公理: $(A \succsim B) \wedge (B \succsim C) \Rightarrow (A \succsim C)$

- 例: あるエージェントが非推移性の選好を持っているなら、お金をすべて失うかもしれない
  - 仮に B > C ならば, このエージェントはBを得るためにCに加えて、少しのお金を払う（1円でも）
  - 仮に A > B ならば, このエージェントはAを得るためにAに加えて、少しのお金を払う（1円でも）
  - 仮に C > A ならば、このエージェントはCを得るためにAに加えて、少しのお金を払う（1円でも）



合理的な選好

合理性の公理

<u>順序可能性</u>（エージェントは、必ず以下のいずれかの選好を持つ）

$$(A \succ B) \vee (B \succ A) \vee (A \sim B)$$

<u>推移性</u>

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$$

<u>継続性</u>

$$A \succ B \succ C \Rightarrow \exists p\ [p, A;\ 1-p, C] \sim B$$

<u>代替性</u>

$$A \sim B \Rightarrow [p, A;\ 1-p, C] \sim [p, B; 1-p, C]$$

<u>単調性</u>

$$A \succ B \Rightarrow$$
$$(p \geq q \Leftrightarrow [p, A;\ 1-p, B] \succeq [q, A;\ 1-q, B])$$

合理的な選好がある

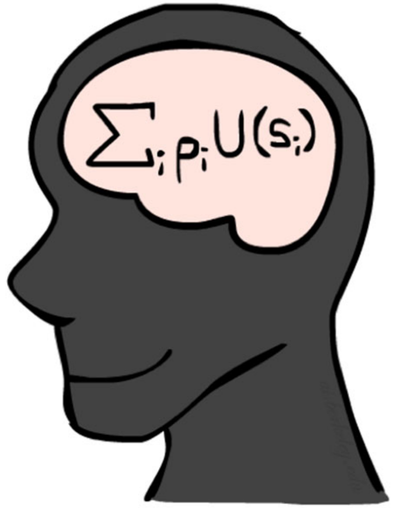
→期待効用を最大化するものとして説明できるような行動が起きる

期待効用の最大化の原則

- 公理 [Ramsey, 1931; von Neumann & Morgenstern, 1944]
  - 先ほどのスライドの制約をみたす選好が存在するとき、実数値をとる関数Uが以下のような形で存在する：

$$U(A) \geq U(B) \;\Leftrightarrow\; A \succeq B$$

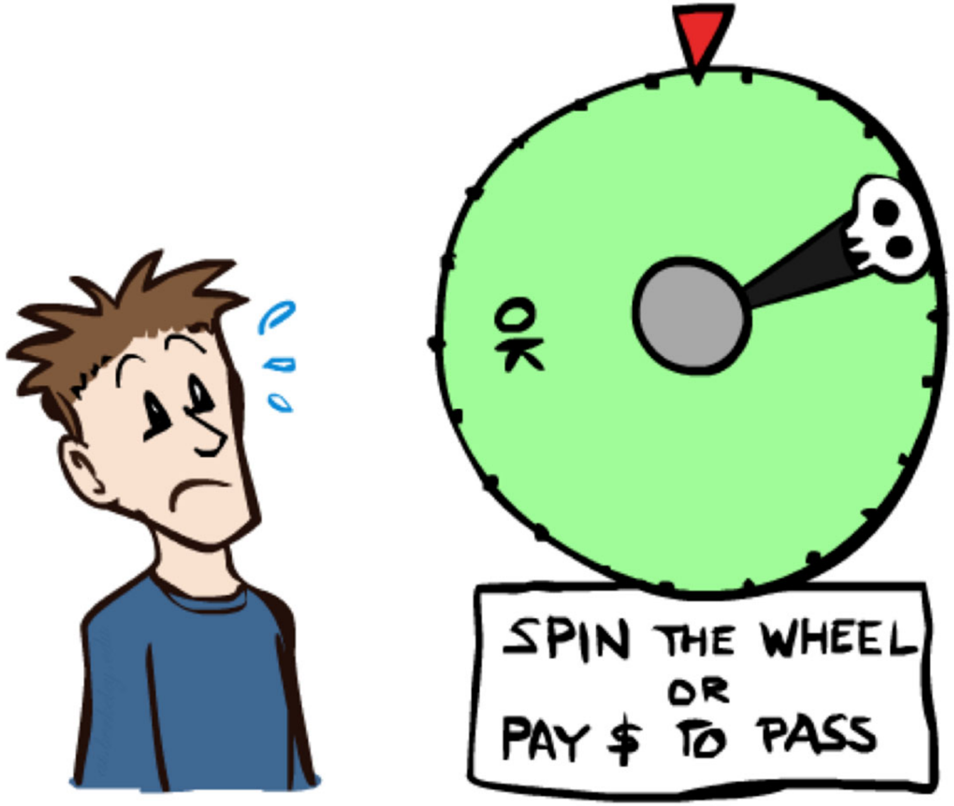
$$U([p_1, S_1;\; \ldots\; ;\; p_n, S_n]) = \sum_i p_i U(S_i)$$



  - つまり、Uは個別の賞品の良さ（どっちを好むか）だけでなく、くじ引きのような場合の期待値（どっちのくじ引きを好むか）を表現できる
- 期待効用の最大化(MEU: Maximum expected utility)の原則:
  - 期待効用を最大化するような行動を選ぶ
  - 注: 効用と確率を直接的に表現したり操作したりしない場合でも、エージェントは合理的な行動をしうる（期待効用の最大化の原則をみたしうる）

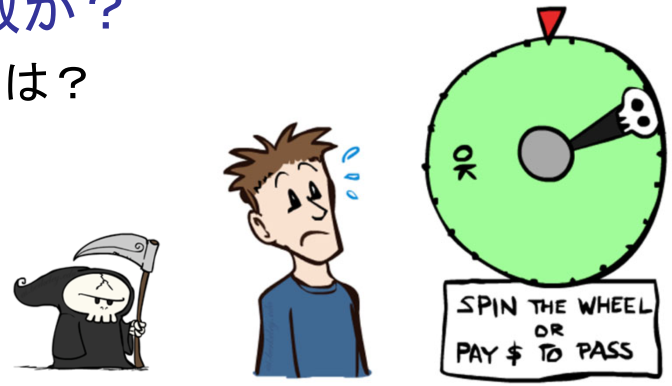
  → 例：最適な戦略が書かれたルックアップテーブルを参照して３目並べ(tic-tac-toe)をする

人間の場合の効用

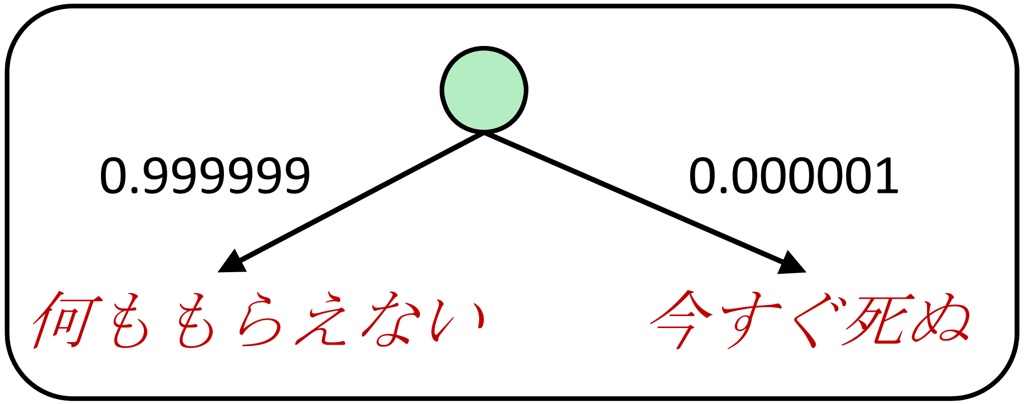


人間の場合の効用

- 効用は、状態を実数に対応付ける．では、どのような数か？
  - どちらを好むか（順序効用）ではなく、数を本当に取り出すには？
- 人間の効用を見積もる標準的なアプローチ：
  - ある賞品A と標準化したくじ引き L_p を比較
    - 確率pで得られる良い賞品 u_+
    - (1-p)の確率で最悪の結末 u_-
  - 両者が等価($A \sim L_p$)になるようにくじ引きの確率pを調整
  - 結果として得られるpは、[0,1]の値をとる効用



3000円払う ～



様々な効用の評価指標

- 正規化された効用: $u_+ = 1.0$, $u_- = 0.0$
- マイクロモート(Micromorts): 100万分の1の可能性での死亡リスク（リスク評価で利用される、リスクを下げるために人々はどれぐらいの額を払うか？）
- 質調整生存年(QALYs): 健康な１年が１単位．健康を損ねた状態の１年は１より小さな値．医療に関する文脈で、生存における質と量を同時に評価する方法.
- 注: 効用を（正の）線形変換しても、行動は不変

$$U'(x) = k_1 U(x) + k_2 \quad \text{where } k_1 > 0$$

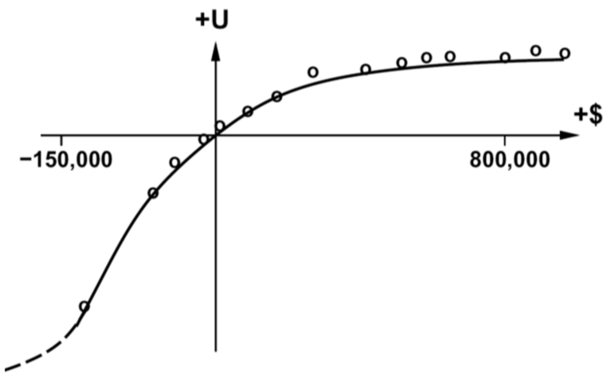
例題: 人間は合理的？

- Allais による有名な例 (1953)
  - A: [0.8, 40万円;　0.2, \$0]
  - B: [1.0, 30万円;　0.0, \$0]
  - C: [0.2, 40万円;　0.8, \$0]
  - D: [0.25, 30万円;　0.75, \$0]
- 多くの人は B > A, C > D
- でも、仮に U(0円) = 0 とするならば、
  - B > A ⇒ U(30万円) > 0.8 U(40万円)
  - C > D ⇒

    ⇒

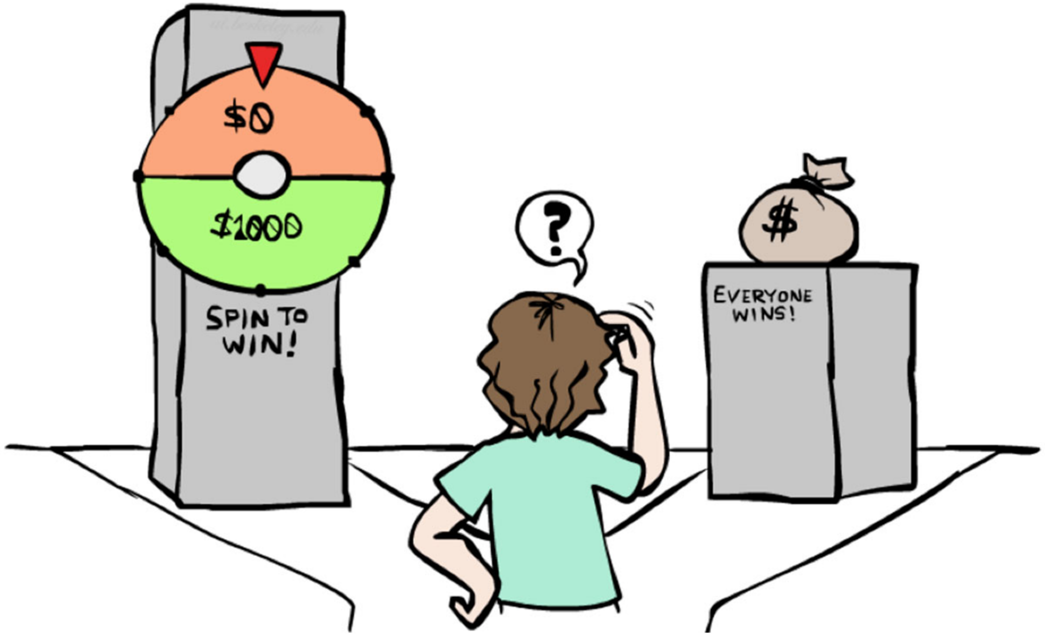
現金

- 現金は効用関数にはならない．現金（借金）を持つことの効用を議論することはできる．
- くじ引き L = [p, $X; (1-p), $Y] があるとき、
  - 現金での期待値 EMV(L) は p*X + (1-p)*Y
  - U(L) = p*U($X) + (1-p)*U($Y)
  - 典型的には U(L) < U(EMV(L))
    （くじ引きよりも、期待額を現金でもらうことを好む）
  - この意味では、人々はリスクを避ける
  - 借金がかさんだ時には、人々はリスクを取りがち



保険の例

- くじ引き [0.5, 10万円; 0.5, 0円]
  - 現金での期待値は (5万円)
  - これの確実性等価(certainty equivalent)は？
    - このくじ引きを引くために許容する支払額
    - 多くの人は4万円
  - この差額の1万円 は保険の付加価値
    - 人々がリスクを減らすために支払うので保険業界が存在できる
    - もし誰もがリスクに対して中立的な立場であれば保険は必要ない
  - これはwin-winの関係: 人々は4万円 を持っておきたいし、保険会社はくじ引きをしたい (彼らの効用曲線はフラットであり、ゆえに、多くのくじ引きをしたがる)



参考

効用関数の応用例

[Morales et al., 2012]

- 「並んで歩く」行動

参考

効用関数の応用例

[Morales et al., 2012]

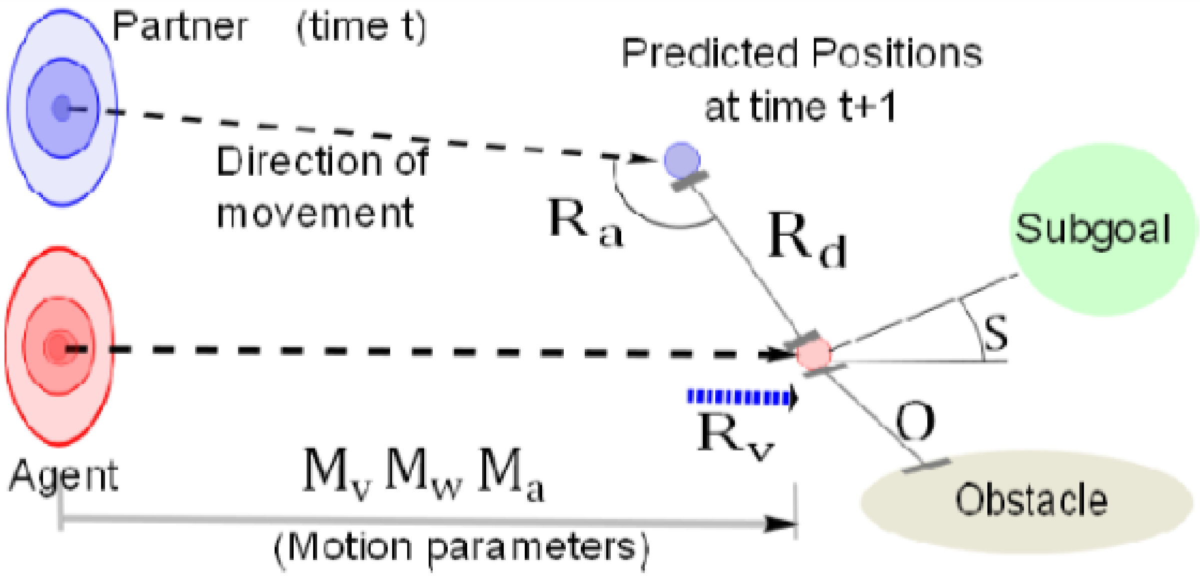
- 失敗例：　人の横に行く、という効用だけだと

参考

効用関数の応用例

[Morales et al., 2012]

$$U(p^i, p^j) = k_O \cdot f_O + k_S \cdot f_S + k_{R_d} \cdot f_{R_d} + k_{R_a} \cdot f_{R_a} + k_{R_v} \cdot f_{R_v} + k_{M_a} \cdot f_{M_a} + k_{M_v} \cdot f_{M_v} + k_{M_w} \cdot f_{M_w}$$



参考

[Morales et al., 2012]

効用関数の応用例

High Utility
Low Utility
partner self
anticipation

The robot anticipates where the partner person will likely to go, and plan its future trajectory

Y. Morales, et al.,, How Do People Walk Side-by-Side? – Using a Computational Model of Human Behavior for a Social Robot, *HRI 2012*

まとめ

- （結果の）不確かさ
  - 環境要因、（敵対的な行動ではない）相手の動き
→確率的な事象としてモデル化
  - 期待マックス法　⇔　ミニマックス法
- 効用
  - 評価関数：順序性が問題
  - 効用関数：期待値が正しいことが必要
  - 合理性：期待効用を最大化するような行動

引用文献

- S. Russell and P. Norvig, Artificial Intelligence A Modern Approach, 3rd edition, Pearson Education Limited, 2016.
- UC Berkeley CS188 Intro to AI -- Course Materials http://ai.berkeley.edu/
- S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, The MIT Press, 2005.
- The Web site for the text "Probabilistic Robotics" http://www.probabilistic-robotics.org/