Learning to play Pac-Man

Christian Buschmann
Artificial Intelligence for Games

Overview

- Pac-Man
  - The Game
  - Game Playing Agent
  - Evolutionary Approach
  - Conclusions
- Ms. Pac-Man
  - The Game
  - Game Playing Agent
  - Action Modules
  - Policy Learning
  - Conclusions

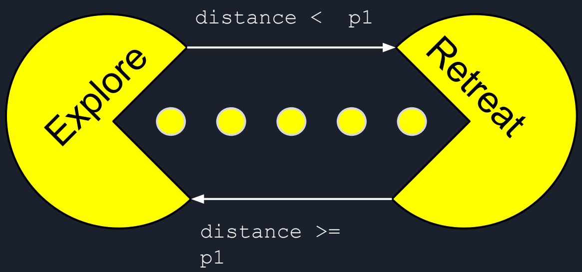
Pac-Man

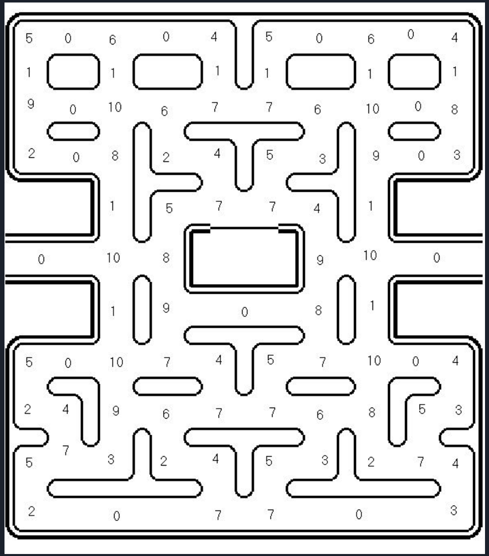
Pac-Man - The Game

- One player controls “Pac-Man” through a maze
  - Pac-Man constantly moves, player controls direction
  - Left and right side walls have “warp” exits
- Maze is filled with dots (points) and power pills
  - Fruits spawn for short periods of time
  - Provide many points
- Player is chased by four ghosts with predetermined behaviours
  - 3 modes, one of which is random
- Eating a power pill lets player eat ghosts
  - ghosts turn blue and try to run from player
  - killed ghosts return after short wait in center of room
- Goal: collect all dots without losing last life

Pac-Man - Game Playing Agent

- Simplify game to train effective agent
  - One ghost, no power pills
- Model agent as state machine
  - Transitions defined by distance to ghost
- Model all moves as distinct turn types





Pac-Man - Game Playing Agent

- Behaviour of agent in each state modeled by 85 parameters
  - 1 parameter for distance to ghost threshold
  - 17 parameters for behaviour likelihood in "Explore" state
  - 76 parameters for behaviour likelihood in "Retreat" state
- Ghost position mapped to 8 "cases"
  - back
  - back-left
  - back-right
  - forward
  - forward-left
  - forward-right
  - left
  - right

Parameter	Description
p_1	Distance to ghost
Explore:	
p_{2-3}	Corridor: forward, backward
p_{4-5}	L-turn: forward, backward
p_{6-8}	T-turn (a) approach centre
p_{9-11}	T-turn (b) approach left
p_{12-14}	T-turn (c) approach right
p_{15-18}	Intersection
Retreat:	
p_{19-20}	Corridor: ghost forward
p_{21-22}	Corridor: ghost behind
p_{23-24}	L-turn: ghost forward
p_{25-26}	L-turn: ghost behind
p_{27-29}	T-turn (a): ghost behind
p_{30-32}	T-turn (b): ghost behind
p_{33-35}	T-turn (c): ghost behind
p_{36-38}	T-turn (a): ghost on left
p_{39-41}	T-turn (b): ghost on left
p_{42-44}	T-turn (a): ghost on right
p_{45-47}	T-turn (b): ghost on right
p_{48-50}	T-turn (b): ghost forward
p_{51-53}	T-turn (c): ghost forward
p_{54-57}	Intersection : ghost forward
p_{58-61}	Intersection : ghost behind
p_{62-65}	Intersection : ghost left
p_{66-69}	Intersection : ghost right
p_{70-73}	Intersection : ghost forward/left
p_{74-77}	Intersection : ghost forward/right
p_{78-81}	Intersection : ghost behind/left
p_{82-85}	Intersection : ghost behind/right

Pac-Man
Evolutionary Approach

Pac-Man - Evolutionary Approach

- Agent behaviour depends entirely on 85 parameter vector
  - Stochastic movement
- Agent can be improved via genetic algorithm applied on this vector
- Fitness function:

$$f = \sum_{level} \frac{score_{level}}{maxscore_{level}} + \min\left\{\frac{time_{level}}{maxtime_{level}}, 1\right\}$$

- Each instance runs 10 times due to stochastic movement

Pac-Man - Evolutionary Approach

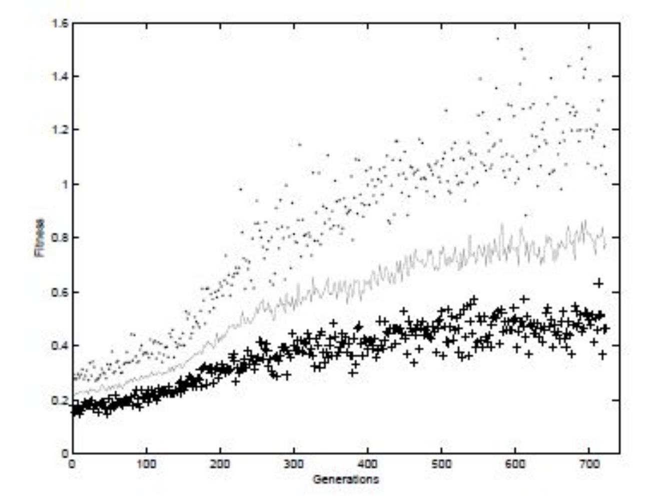
- Setup of agent allows for hand-coding of parameters
  - Allows for manual experimentation
- Three manual parameter sets:
  - P_{h1} : equal probabilities for each parameter
  - P_{h2} : less likely to turn around, never moves towards ghost during “retreat”
  - P_{h3} : very unlikely to turn around

Vector	Mean	Std. Dev.	Min.	Max.
P_{h1}	0.215	0.080	0.104	0.511
P_{h2}	0.613	0.316	0.187	1.699
P_{h3}	1.372	0.522	0.276	2.062

- Limitations of agent become visible
  - No knowledge of points in maze
  - Very rough estimate of ghost position

Pac-Man - Evolutionary Approach

- PBIL used for evolution
- 250 games per generation
  - Population of 25
  - 10 games per parameter set
- End results above P_{h1} and P_{h2}
- Still slightly worse than P_{h3}
- Parameters converge to similar values



Pac-Man - Conclusions

- Limitations of simple rule-based agent clear
- Parameter bloat
- Lacking “intelligence”
- Extending on such a simple rule set based representation impractical
- Useful as benchmark, not as playing agent

Ms. Pac-Man

Ms. Pac-Man - The Game

- Variation of regular Pac-Man
- Different levels
  - 2 extra “warp” exits
  - Fruits more random
- Ghosts don’t strictly follow set behaviour patterns
  - Different base behaviour
  - Randomness factor added

Ms. Pac-Man - Game Playing Agent

- Agent can be defined by set of rules
  - Includes tie-breaking mechanism
- Rules are human-readable
  - Easy to include domain knowledge
- Action modules containing conditions, observations, and actions
  - Determine behaviour of agent
- Requirements for rule-based approach:
  - Possible actions
  - Possible conditions
  - How to make rules from conditions and actions
  - How to combine rules into policies

Ms. Pac-Man Action Modules

Ms. Pac-Man - Action Modules

- Modules ranked by priority
- Every module can be switched “on” or “off”
  - Agent can use any subset of modules
- Highest ranked module determines direction
- Tie-breaker for equally ranked directions
  - Next highest ranked direction decides
  - If no tie-breaker possible, choose randomly
- Decisions made each full grid cell
  - ca. 25 game ticks / 0.2 seconds

Ms. Pac-Man - Action Modules

- Easy to manually implement actions
  - Induce domain knowledge
- Actions not exclusive
- Each module assigned a priority
  - Priority needs to be learned

Name	Description
ToDot	Go towards the nearest dot.
ToPowerDot	Go towards the nearest power dot.
FromPowerDot	Go in direction opposite to the nearest power dot.
ToEdGhost	Go towards the nearest edible (blue) ghost.
FromGhost	Go in direction opposite to the nearest ghost.
ToSafeJunction	Go towards the maximally safe junction. For all four directions, the "safety" of the nearest junction is estimated in that direction. If Ms. Pac-Man is n steps away from the junction and the nearest ghost is k steps away, then the safety value of this junction is $n - k$. A negative value means that Ms. Pac-Man possibly cannot reach that junction.
FromGhostCenter	Go in a direction which maximizes the Euclidean distance from the geometrical center of ghosts.
KeepDirection	Go further in the current direction, or choose a random available action (except turning back) if that is impossible.
ToLowerGhostDensity	go in the direction where the cumulative ghost density decreases fastest. Each ghost defines a density cloud (with radius = 10 and linear decay), from which the cumulative ghost density is calculated.
ToGhostFreeArea	Choose a location on the board where the minimum ghost distance is largest, and head towards it on the shortest path.

Ms. Pac-Man - Action Modules

- Set of observations required to build rules
  - e.g. distances to objects
- Manually defined
  - Can be improved
  - Good baseline
- Default to maximum value if unknown
- Easily calculated by agent

Name	Description
`Constant`	Constant 1 value.
`NearestDot`	Distance of nearest dot.
`NearestPowerDot`	Distance of nearest power dot.
`NearestGhost`	Distance of nearest ghost.
`NearestEdGhost`	Distance of nearest edible (blue) ghost.
`MaxJunctionSafety`	For all four directions, the "safety" of the nearest junction in that direction is estimated, as defined in the description of action `ToSafeJunction`. The observation returns the value of the maximally safe junction.
`GhostCenterDist`	Euclidean distance from the geometrical center of ghosts.
`DotCenterDist`	Euclidean distance from the geometrical center of uneaten dots.
`GhostDensity`	Each ghost defines a density cloud (with radius = 10 and linear decay). Returns the value of the cumulative ghost density.
`TotalDistToGhosts`	"travelling salesman distance to ghosts:" the length of the shortest route that starts at Ms. Pac-Man and reaches all four ghosts (not considering their movement).

Ms. Pac-Man - Action Modules

- Conditions made up of observations
  - Joined with logic operators
- Example condition:

(NearestDot<5) and (NearestGhost>8) and (FromGhost+)

- Rules constructed from condition and action
- Example rule:

if (NearestDot<5) and (NearestGhost>8) and (FromGhost+) then FromGhostCenter+

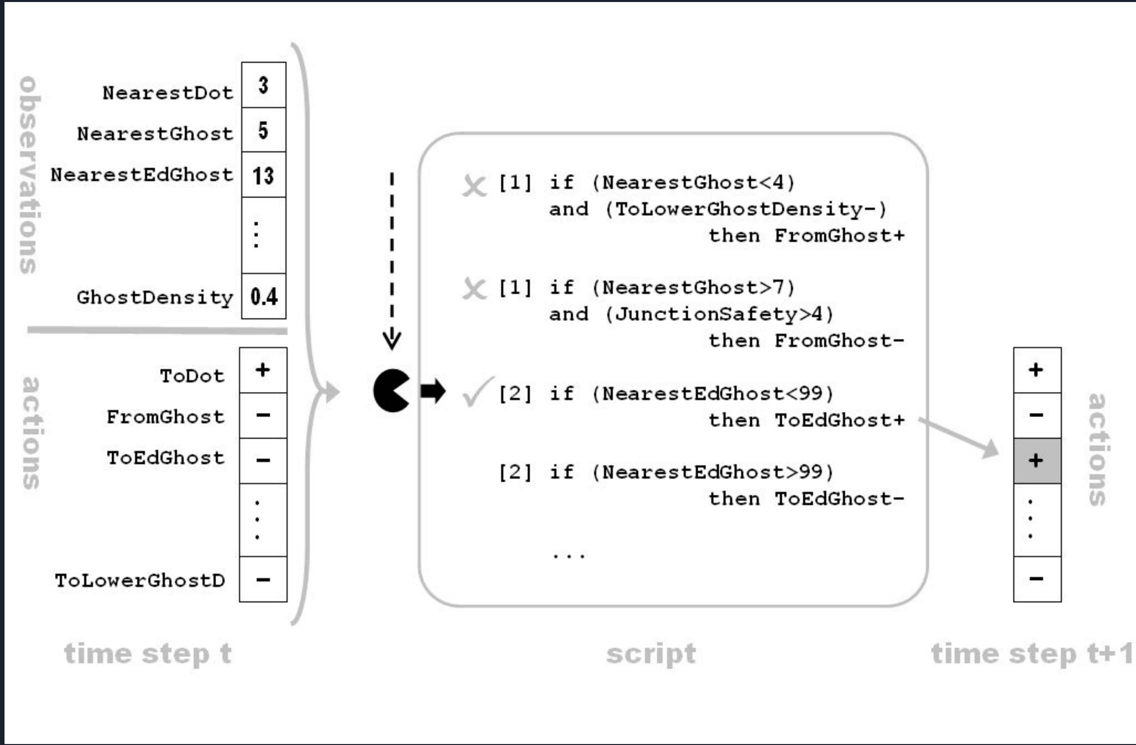
Ms. Pac-Man - Action Modules

- Action modules combine into policies
  - Example hand-coded policy:

```
[1] if NearestGhost<4 then FromGhost+
[1] if NearestGhost>7 and JunctionSafety>4 then FromGhost-
[2] if NearestEdGhost>99 then ToEdGhost-
[2] if NearestEdGhost<99 then ToEdGhost+
[3] if Constant>0 then KeepDirection+
[3] if FromPowerDot- then ToPowerDot+
[3] if GhostDensity<1.5 and NearestPowerDot<5 then FromPowerDot+
[3] if NearestPowerDot>10 then FromPowerDot-
```

- Rules stay switched on until explicitly switched off or replaced

Ms. Pac-Man - Action Modules



Learning to play Pac-Man

Ms. Pac-Man Policy Learning

Ms. Pac-Man - Policy Learning

- Requirements for learning:
  - Set of rules
  - Set of rule slots (policy)
- Each rule slot has a priority
- Each rule slot has a probability p_i to contain a rule
  - Each rule picked with probability $q_{i,j}$
- Probabilities for filling slots learned by algorithm

Ms. Pac-Man - Policy Learning

- Ruleset can be generated instead of predefined
- Randomly pick 2 conditions
  - Values picked uniformly from set for each condition module
- Randomly pick one action module
  - 50% chance to turn it on or off

Ms. Pac-Man - Conclusions

- Random Rule-sets
  - Cross-Entropy Method compared to Stochastic Gradient
  - 100 rules
  - 100 rule slots
- Hand-coded Rule-set
  - 42 rules
  - 30 rule slots
- Baseline comparisons
  - Random policy of 10 rules
  - Hand-coded policy

Method	Avg. Score	(25%/75% percentiles)
CE-RANDOMRB	6382	(6147/6451)
CE-FIXEDRB	**8186**	(6682/9369)
SG-RANDOMRB	4135	(3356/5233)
SG-FIXEDRB	5449	(4843/6090)
CE-RANDOMRB-1ACTION	5417	(5319/5914)
CE-FIXEDRB-1ACTION	5631	(5705/5982)[6]
SG-RANDOMRB-1ACTION	2267	(1770/2694)
SG-FIXEDRB-1ACTION	4415	(3835/5364)
Random policy	676	(140/940)
Hand-coded policy	7547	(6190/9045)
Human play	8064	(5700/10665)

Ms. Pac-Man - Conclusions

- Best policy learned by fixed rule base:

```
[1] if NearestGhost<3 then FromGhost+
[1] if MaxJunctionSafety>3 then FromGhost-
[2] if NearestEdGhost>99 then ToPowerDot+
[2] if NearestEdGhost<99 then ToEdGhost+
[2] if GhostDensity<1.5 and NearestPowerDot<5 then FromPowerDot+
[3] if Constant>0 then ToCenterofDots+
```

- Random rule-set policies behave similarly to fixed rule-set

Ms. Pac-Man - Conclusions

- Best policy learned by random rule base:

```
[1] if MaxJunctionSafety>2.5 and ToLowerGhostDensity- then FromGhost-
[1] if NearestGhost<6 and MaxJunctionSafety<1 then FromGhost+
[1] if NearestGhost>6 and FromGhostCenter- then ToEdGhost+
[2] if ToEdGhost- and CenterOfDots>20 then ToEdGhost+
[2] if ToEdGhost- and NearestEdGhost<99 then ToEdGhost+
[2] if NearestDot>1 and GhostCenterDist>0 then KeepDirection+
[3] if ToGhostFreeArea- and ToDot- then ToPowerDot+
```

- Contains superfluous rules

Ms. Pac-Man - Conclusions

- Ability to perform multiple actions concurrently is essential
- CEM performs better than SG
  - Could be fixed with thorough search over parameter space
  - CEM reaches good play faster
- No agent evolved tactic of “luring ghosts in”
- Time-related conditions lacking