ENVISIM COM NEAT

Daniel Fusimoto Pires RA:11201921874

NEAT (NeuroEvolution of Augmenting Topologies)

- É uma abordagem de criação de rede neural, onde começamos apenas com a layer de neurônios de entrada e a layer de neurônios de saída, sem hidden layers.
- A partir desse estado inicial, o algoritmo NEAT decide como deve ser a arquitetura interna da rede:
  - Adiciona e remove ligações
  - Adiciona e remove layers
  - Muda pesos das ligações

População, Espécies e Genomas

Para descobrir a melhor arquitetura para o problema, o NEAT gera um conjunto de redes neurais (população) com arquiteturas diferentes (genomas), inicialmente todas parecidas (mesma espécie).

Depois de cada iteração, precisamos fornecer uma nota (fitness) para cada genoma. Baseado nessa nota, o NEAT separa os melhores genomas, gera mais cópias deles e realiza o processo de mutação, alterando aleatoriamente parte da arquitetura dos genomas. Até mesmo juntando partes de dois genomas diferentes (crossing-over).

ENVISIM?

Para conseguir rodar o NEAT, era necessário várias instâncias do EnviSim, algo impossível no jogo original.

Foi feita uma reescrita simplificada do funcionamento do EnviSim para rodar no terminal, usando emojis como display das casas.

Além disso, com a ajuda do Gustado Del Rio, foi implementado um gerador aleatório de mapas.

Abordagem do problema

Devido a complexidade do problema, ele foi dividido em duas etapas de treinamento:

- 1ª Ensinar a não morrer, a pegar o ouro, e a sair do mapa
  - Cenários simples, e com o personagem já de frente para o problema
  - Dado o cenário, a rede gerava um output, e no geral, para cada output inteligente, ganhava 1 de fitness
  - Alguns cenários mais importantes ganhavam mais fitness ao acertar
    - Pegar o ouro
    - Sair do mapa

Abordagem do problema

Quando um genoma passava em todos os cenários, o que neste caso representa um fitness de 47, finaliza-se a primeira etapa de treinamento:

```
****** Running generation 1965 ******

Population's average fitness: 34.26633 stdev: 8.34220
Best fitness: 47.00000 - size: (6, 43) - species 3 - id 2285
Average adjusted fitness: 0.648
Mean genetic distance 2.525, standard deviation 0.774
Population of 200 members in 4 species:
   ID   age  size  fitness  adj fit  stag
  ====  ===  ====  =======  =======  ====
     2  1251    50     46.0    0.703   787
     3  817     47     47.0    0.588   786
    33   37     46     44.0    0.671    10
    34   10     57     42.0    0.630     8
Total extinctions: 0
Generation time: 0.260 sec (0.235 average)
```

- 2ª Ensinar a andar em um mapa

Como a rede neural já sabe como lidar com os obstáculos, passamos para treiná-la a como completar um mapa. No final da jogada, para cada conquista alcançada, damos um fitness:

```
def avaliate_game(reached, grabbed, win, dead, steppedOnFlash, reachedExit, dumbness):
    fitness = 0
    if steppedOnFlash:
        fitness += 5
    if reached:
        fitness += 10
    if grabbed:
        fitness += 20
    if reachedExit:
        fitness += 30
    if win:
        fitness += 100
    if dead:
        fitness -= 10

    fitness -= dumbness
    return fitness
```

Nessa etapa foram implementadas algumas penalidades para conseguir filtrar melhor os melhores genes, dentre elas:

```
# Penalidades e fim de jogo
if reachedExit and not win:
    break
if map[pos] == 4:
    reachedGoal = True
if map[pos] == 3 or map[pos] == 8 or map[pos] == 9 or map[pos] == 11:
    steppedOnFlash = True
if (vector[0][6] and command == 3) or command_memory[-3:] == [11, 11, 11] or command_memory[-3:] == [12, 12, 12] or
command_memory[-2:] == [13, 13] or command_memory[-2:] == [11, 12] or command_memory[-2:] == [12, 11]:
    dumbness += 1
if (command == 0 and map[pos] != 4) or (command == 0 and map[pos] == 4 and grabbed):
    dumbness += 1
if (command == 1 and map[pos] != 5) or (command == 1 and not grabbed):
    dumbness += 1
if len(pos_memory) >= 3 and pos_memory[-3][0] == pos_memory[-2][0] == pos_memory[-1][0]:
    dumbness += 1
if grabbed:
    if len(pos_memory) > 0 and map[pos_memory[0][0]] != 4:
        pos_memory = []
    if len(pos_memory) >= 3 and pos_memory[-3][0] == pos_memory[-2][0] == pos_memory[-1][0] and map[pos] == 4:
        dumbness += 1
    if map[pos] == 5:
        reachedExit = True
```

Quando uma rede chega num fitness bom, que para o caso desta implementação é por volta de 200, podemos parar a evolução e gerar um arquivo referente a essa rede.

```
Population's average fitness: -404.77500 stdev: 562.16780
Best fitness: 202.00000 - size: (6, 39) - species 40 - id 6831
Average adjusted fitness: 0.619
Mean genetic distance 2.583, standard deviation 0.844
Population of 197 members in 3 species:
   ID   age  size  fitness  adj fit  stag
  ====  ===  ====  =======  =======  ====
    36   52    60    200.0    0.553    13
    39   26    83     71.0    0.676     8
    40   21    54    202.0    0.627    12
Total extinctions: 0
Generation time: 1.140 sec
Saving checkpoint to neat-checkpoint-2050
```