

Thi đấu

Chọn ngẫu nhiên một đôi cá thể sử dụng phân bố xác suất đều, sau đó chọn cá thể tốt hơn trong hai cá thể đó như một cá thể cha mẹ. Thực hiện tương tự để chọn cá thể cha mẹ còn lại.

Lựa chọn cá thể tinh hoa

Các phương pháp trên đều không đảm bảo cá thể tốt nhất được lựa chọn. Trong phương pháp lựa chọn tinh hoa, một số lượng nhất định các cá thể tốt nhất được lựa chọn trước, sau đó phần còn lại được lựa chọn theo các phương pháp ru lết hay thi đấu như ở trên. Như vậy, các cá thể tốt nhất luôn được duy trì đoạn gen của mình sang thế hệ sau và tránh làm mất lời giải tốt nhất đã tìm được.

Giá trị xác suất

Xác suất lai ghép. Xác suất lai ghép được lựa chọn tương đối lớn, thường từ 0.5 trở lên.

Xác suất đột biến. Xác suất đột biến được lựa chọn rất nhỏ, ít khi vượt quá 0.1. Xác suất đột biến nhỏ để tránh cho thuật toán di chuyển theo kiểu ngẫu nhiên.

Huấn luyện mạng nơ ron để chơi trò chơi "Flappy Bird"

1. Tạo ra 100 cá thể chim với mạng nơ ron riêng cho mỗi cá thể.
2. Khởi tạo ngẫu nhiên bias và weight cho mạng nơ ron.
3. Kiểm thử và đánh giá mạng nơ ron dựa trên quãng đường chim vượt qua được mà không bị va chạm.
4. Lựa chọn những cá thể có hiệu suất tốt nhất dựa trên tiêu chí đánh giá.
5. Áp dụng quy luật sinh tồn như lai ghép và đột biến để tạo ra thế hệ tiếp theo.
6. Lặp lại quá trình từ bước 3 đến bước 5 để cải thiện hiệu suất của các cá thể.

Về game flappy Bird

Qua các lần lặp, hi vọng rằng mỗi thế hệ chim mới sẽ cải thiện hiệu suất so với thế hệ trước đó. Quá trình này tiếp tục cho đến khi thu được một "chú chim bất khả chiến bại" với hiệu suất cao và khả năng chơi trò chơi tốt nhất.

Lập trình

1. Lập trình trò chơi Flappy Bird Load các dữ liệu, thư viện cần thiết và phông chữ của trò chơi:

```
1 import pygame
2 import random
3 import os
4 import neat
5
6 pygame.font.init()
7
8 WIN_WIDTH = 600
9 WIN_HEIGHT = 800
10 FLOOR = 730
11 STAT_FONT = pygame.font.SysFont("comicsans", 50)
12 END_FONT = pygame.font.SysFont("comicsans", 70)
13 DRAW_LINES = False
14
15 WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
16 pygame.display.set_caption("Flappy Bird")
17
18 pipe_img = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "pipe.png")).convert_alpha())
19 bg_img = pygame.transform.scale(pygame.image.load(os.path.join("imgs", "bg.png")).convert_alpha(), (600, 900))
20 bird_images = [pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird" + str(x) + ".png"))) for x in range(1,4)]
21 base_img = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "base.png")).convert_alpha())
22 gen = 0
```

Lớp Bird

```
class Bird:  
    """  
    Bird class representing the flappy bird  
    """  
  
    MAX_ROTATION = 25  
    IMGS = bird_images|  
    ROT_VEL = 20  
    ANIMATION_TIME = 5  
  
    def __init__(self, x, y):  
        """  
        Initialize the object  
        :param x: starting x pos (int)  
        :param y: starting y pos (int)  
        :return: None  
        """  
  
        self.x = x  
        self.y = y  
        self.tilt = 0 # degrees to tilt  
        self.tick_count = 0  
        self.vel = 0  
        self.height = self.y  
        self.img_count = 0  
        self.img = self.IMGS[0]  
  
    def jump(self):  
        """  
        make the bird jump  
        :return: None  
        """  
  
        self.vel = -10.5  
        self.tick_count = 0  
        self.height = self.y
```

Lớp Pipe

```
class Pipe():
    """
    represents a pipe object
    """

    GAP = 200
    VEL = 5

    def __init__(self, x):
        """
        initialize pipe object
        :param x: int
        :param y: int
        :return: None
        """
        self.x = x
        self.height = 0

        # where the top and bottom of the pipe is
        self.top = 0
        self.bottom = 0

        self.PIPE_TOP = pygame.transform.flip(pipe_img, False, True)
        self.PIPE_BOTTOM = pipe_img

        self.passed = False

        self.set_height()

    def set_height(self):
        """
        set the height of the pipe, from the top of the screen
        :return: None
        """
        self.height = random.randrange(50, 450)
        self.top = self.height - self.PIPE_TOP.get_height()
        self.bottom = self.height + self.GAP
```

Lớp Base

```
class Base:
    VEL = 5
    WIDTH = base_img.get_width()
    IMG = base_img

    def __init__(self, y):
        """
        Initialize the object
        :param y: int
        :return: None
        """

        self.y = y
        self.x1 = 0
        self.x2 = self.WIDTH

    def move(self):
        self.x1 -= self.VEL
        self.x2 -= self.VEL
        if self.x1 + self.WIDTH < 0:
            self.x1 = self.x2 + self.WIDTH

        if self.x2 + self.WIDTH < 0:
            self.x2 = self.x1 + self.WIDTH

    def draw(self, win):
        win.blit(self.IMG, (self.x1, self.y))
        win.blit(self.IMG, (self.x2, self.y))
```

```
def run(config_file):

    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                                neat.DefaultSpeciesSet, neat.DefaultStagnation,
                                config_file)

    p = neat.Population(config)

    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)

    winner = p.run(eval_genomes, 50)

    print('\nBest genome:\n{}'.format(winner))
```

```
def eval_genomes(genomes, config):
    """
    runs the simulation of the current population of
    birds and sets their fitness based on the distance they
    reach in the game.
    """
    global WIN, gen
    win = WIN
    gen += 1

    nets = []
    birds = []
    ge = []
    for genome_id, genome in genomes:
        genome.fitness = 0
        net = neat.nn.FeedForwardNetwork.create(genome, config)
        nets.append(net)
        birds.append(Bird(230, 350))
        ge.append(genome)

    base = Base(FLOOR)
    pipes = [Pipe(700)]
    score = 0

    clock = pygame.time.Clock()

    run = True
    while run and len(birds) > 0:
        clock.tick(30)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()
                quit()
                break
```

```
pipe_ind = 0
if len(birds) > 0:
    if len(pipes) > 1 and birds[0].x > pipes[0].x + pipes[0].PIPE_TOP.get_width():
        pipe_ind = 1

for x, bird in enumerate(birds):
    ge[x].fitness += 0.1
    bird.move()

    output = nets[birds.index(bird)].activate((bird.y, abs(bird.y - pipes[pipe_ind].height), abs(bird.y - pipes[pipe_ind].PIPE_BOTTOM.get_y()), bird.jumped))
    if output[0] > 0.5:
        bird.jump()

base.move()

rem = []
add_pipe = False
for pipe in pipes:
    pipe.move()
    for bird in birds:
        if pipe.collide(bird, win):
            ge[birds.index(bird)].fitness -= 1
            nets.pop(birds.index(bird))
            ge.pop(birds.index(bird))
            birds.pop(birds.index(bird))

    if pipe.x + pipe.PIPE_TOP.get_width() < 0:
        rem.append(pipe)

    if not pipe.passed and pipe.x < bird.x:
        pipe.passed = True
        add_pipe = True
```

te odio. Sed
elementum
id aliquet
na ferment-
tellus cras
is ut diam
iam phasel-
met dictum

er sit amet
eugiat nibh
t duis tris-
auris nunc
i fames ac
lch mauris.
At

```
for x, bird in enumerate(birds):
    ge[x].fitness += 0.1
    bird.move()

    output = nets[birds.index(bird)].activate((bird.y, abs(bird.y - pipes[pipe_ind].height), abs(bird.y - pipes[pipe_index].bottom)))
    if output[0] > 0.5:
        bird.jump()
```

```
add_pipe = False
for pipe in pipes:
    pipe.move()
    for bird in birds:
        if pipe.collide(bird, win):
            ge[birds.index(bird)].fitness -= 1
            nets.pop(birds.index(bird))
            ge.pop(birds.index(bird))
            birds.pop(birds.index(bird))

        if pipe.x + pipe.PIPE_TOP.get_width() < 0:
            rem.append(pipe)

        if not pipe.passed and pipe.x < bird.x:
            pipe.passed = True
            add_pipe = True

    if add_pipe:
        score += 1
        for genome in ge:
            genome.fitness += 5
        pipes.append(Pipe(WIN_WIDTH))

    for r in rem:
        pipes.remove(r)
```

```
if add_pipe:
    score += 1
    for genome in ge:
        genome.fitness += 5
    pipes.append(Pipe(WIN_WIDTH))

    for r in rem:
        pipes.remove(r)

    for bird in birds:
        if bird.y + bird.img.get_height() - 10 >= FLOOR or bird.y < -50:
            nets.pop(birds.index(bird))
            ge.pop(birds.index(bird))
            birds.pop(birds.index(bird))

draw_window(WIN, birds, pipes, base, score, gen, pipe_ind)
```

