به نام خدا

گزارش درس هوش محاسباتی

موضوع : flappy bird

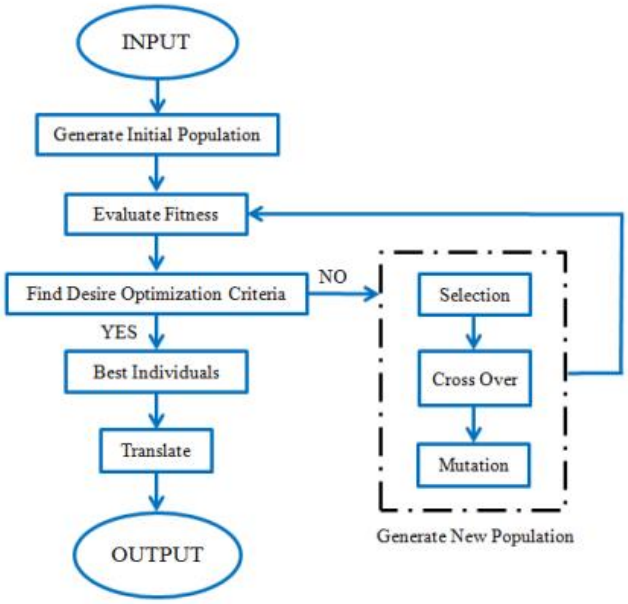
نام استاد : جناب آقای علی تورانی

نام نویسنده : سحر فخریه کاشان

شماره دانشجویی : 960122680012

این پروژه چیست و چگونه کار میکند؟

این پروژه بر اساس الگوریتم ژنتیک است. در ویدیو نحوه عملکرد این الگوریتم توضیح داده شده که به صورت مختصر به آن میپردازیم:



Initializing population : 50 عدد پرنده به صورت رندوم انتخاب میکنیم و هر پرنده شبکه عصبی مخصوص خودش را دارد

Fitness Evaluation : براساس نحوه عملکرد هر پرنده و میزان زنده ماندن آن ها در بازی مقدار برازششان تعیین میگردد که در ادامه نحوه محاسبه آن بیان خواهد شد

Selection : درصدی از بهترین پرندگان (پرندگانی که بیشتر در بازی زنده مانده اند) انتخاب میشوند

Mutation and crossover : در این مرحله پرندگان انتخاب شده نسل جدیدی را تولید میکنند و 50 شئ پرنده جدید براساس آن ها ساخته میشود

پس از این مرحله انقدر 3 مرحله اخیر را تکرار میکنیم تا به نسلی که مورد قبول ماست برسیم (Termination criteria)

Termination criteria : در مرحله ای که پرنده (پرنده ها) توانستند به امتیاز 50 برسند یعنی از 50 تا لوله رد شدند میتوانیم بگوییم که به نسل دلخواهمان رسیدیم و برنامه را به اتمام برسانیم

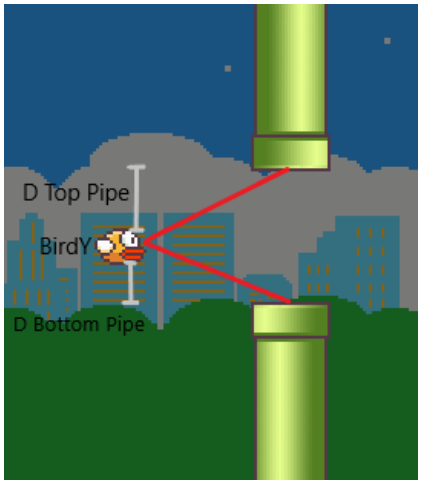
نحوه محاسبه fitness پرنده ها چیست؟

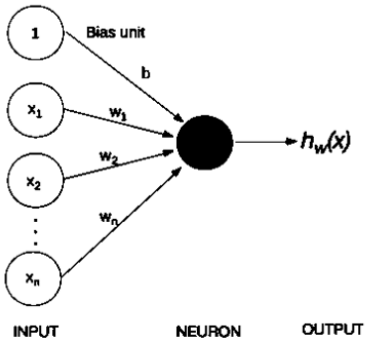
هر پرنده دارای یک شبکه عصبی مخصوص خودش است که در این شبکه، ما برای ورودی :

موقعیت مکانی عمودی پرنده (bird y)، فاصله پرنده تا لوله بالایی و فاصله پرنده تا لوله پایینی را در نظر میگیریم (که هر کدام از این ورودی ها دارای وزن هستند)

و برای خروجی :

تصمیم میگیریم که پرنده بپرد و یا نه؟





$$y_{in} = (1 \times b) + \sum_{i=1}^{n} x_i w_i$$

با وجود ورودی ها ، وزن ها و بایاس میتوانیم با استفاده از فرمول داخل عکس بالا y_{in} را به دست بیاریم سپس آن را به activation finction میدهیم که برابر TanH است که مقداری بین [-1,1] میدهند که اگر مقدار خروجی بیشتر از 0.5 باشد پرنده تصمیم به پرش میگیرد و در غیر این صورت نمیپرد

برای اجرای این الگوریتم از کتابخانه Neat و برای گرافیک آن از کتابخانه Pygame استفاده کردیم در ادامه به نحوه عملکرد این الگوریتم میپردازیم.

Neat چیست؟

NeuroEvolution of Augmenting Topologies (NEAT) یک الگوریتم ژنتیک (GA) برای تولید شبکه های عصبی مصنوعی در حال تکامل (تکنیک تکامل عصبی) است.

Neat چگونه کار میکند ؟

NEAT با یک شبکه feed-forward ، فقط با نورونهای ورودی و خروجی آغاز می شود. هر نسل توسط تولید مثل و جهش مناسب ترین افراد نسل قبلی تولید می شود.

Neat در واقع وزن های رابطه ها(connection) را تغییر میدهد ، رابطه ای جدید را به وجود میاورد یا رابطه ای را حذف میکند و یا نود جدیدی به صورت رندوم تولید میکند و در شبکه قرار میدهد. دلیل این کار پیدا کردن بهترین توپولوژی و ساختار برای حل یک مساله است و اگر لازم باشد پیچیده تر میشود .

در Neat ما باید فایلی به اسم config-feedforward بسازیم و آن را داخل برنامه وارد کنیم. در این فایل تنظیمات اولیه خود را ست میکنیم که در اینجا به مهمترین تنظیمات میپردازیم:

Neat دارای 4 بخش است:

- [NEAT]
- [DefaultStagnation]
- [DefaultReproduction]
- [DefaultGenome]

تنظیمات اساسی بخش [NEAT] :

```
fitness_criterion     = max
```

میتواند مقدار min, max, and mean را بپذیرد. در این حالت ما پرنده های با fitness = max (بهترین پرنده ها) را انتخاب میکنیم.

```
fitness_threshold     = 50
```

وقتی fitness_criterion برابر threshold شود تکامل، خاتمه پیدا میکند در واقع شرط خاتمه‌ی تکامل رسیدن به threshold=50 است

```
pop_size              = 50
```

تعداد پرنده های هر نسل را مشخص میکند

تنظیمات اساسی بخش [DefaultGenome] :

در Neat ما به جمعیتمان ژنوم میگوییم که در این پروژه جمعیت ما ، همان پرنده ها هستند

ژنوم ها دارای نود و ژن هستند که نود ها میتوانند نود های ورودی، خروجی و میانی باشند و ژن ها در واقع ارتباطات (connections) بین این نود ها هستند

هر فرد در جمعیت با ویژگی های تعیین شده در ذیل شروع به کار میکند:

```
activation_default      = tanh
```

تابع فعال سازی خود را در این قسمت مشخص میکنیم.

```
activation_mutate_rate  = 0.0
```

اگر بخواهیم تابع فعال سازی چیزی رندوم باشد در این قسمت درصدش را مشخص میکنیم برای مثال 0.1 = 10%
یعنی وقتی فرد جدیدی وارد جمعیت میشود 10 درصد ممکن است تابع فعال سازی آن متفاوت باشد اما ما در این پروژه میخواهیم فقط از tanh استفاده کنیم پس آن را برابر صفر قرار داده ایم

```
activation_options      = tanh
```

در اینجا گزینه هایی که میخواهیم از آن ها به صورت تابع فعال سازی استفاده کنیم را قرار میدهیم این همان چیزی است که activation_mutate_rate از آن برای پیدا کردن توابع فعال سازی استفاده میکند

```
bias_max_value          = 30.0
```

بیشترین میزان bias که میتواند بگیرد از آنجایی که همه چیز در مرحله اول به صورت رندوم است و ما bias با مقدار بینهایت نمیخواهیم پس مقدار آن را بین [30,30-] در نظر میگیریم

```
bias_min_value          = -30.0
```

کمترین میزان bias که میتواند بگیرد

```
conn_add_prob           = 0.6
conn_delete_prob        = 0.4
```

به این معناست که ما چقدر میخوایم یک رابطه(connection) اضافه و یا حذف شود (که در اینجا 60 درصد اضافه و 40 درصد حذف میشود)

```
enabled_default         = True
```

یعنی رابطه ها (connection) اکتیو هستند و یا نه (true = active)

```
enabled_mutate_rate    = 0.01
```

حدود 1 درصد شانس دارد که دی اکتیو شود.

```
feed_forward         = True
```

از شبکه عصبی feed_forward استفاده میکنیم

```
node_add_prob        = 0.1
node_delete_prob     = 0.1
```

10 درصد شانس دارد که یک نود (نورون) جدید حذف یا اضافه کند

```
num_hidden           = 0
num_inputs           = 3
num_outputs          = 1
```

تعداد نورون های ورودی، خروجی و hidden را میدهیم که در اینجا ما فقط 3 ورودی و 1 خروجی داریم.

تنظیمات اساسی بخش [DefaultStagnation] :

```
species_fitness_func = max
```

از بین انواع مختلف بیشترین fitness را انتخاب خواهیم کرد

```
max_stagnation       = 20
```

یعنی چند نسل میتوانیم جلو برویم بدون این که fitness خود را افزایش دهیم اگر 20 نسل جلو رفتیم و fitness زیاد نشد پس آن نوع را حذف میکنیم

تنظیمات اساسی بخش [DefaultReproduction] :

```
elitism            = 2
```

تعداد افراد متناسب در هر گونه که به همان نسبت از نسلی به نسل دیگر حفظ خواهد شد.

```
survival_threshold = 0.2
```

کسری برای هر گونه مجاز به تولید مثل هر نسل است

* این مقادیر میتوانند تغییر کنند و با تغییر آن ها به جواب های متفاوتی خواهیم رسید

باقی توضیحات این فایل (config) را میتوانید از سایت زیر ببینید :

Configuration file description — NEAT-Python 0.92 documentation (neat-python.readthedocs.io)

برای توضیحات بیشتر در مورد این کتابخانه میتوانید از آدرس زیر استفاده کنید :

NEAT Overview — NEAT-Python 0.92 documentation (neat-python.readthedocs.io)

چگونه این فایل را وارد پروژه خود کنیم؟

```
import neat

# set krdne shabake asabi
Net = neat.nn.FeedForwardNetwork.create(g, config)

def run(config_path):
    # peida krdne sartitrhae lazem
    config = neat.config.Config(
        neat.DefaultGenome,
        neat.DefaultReproduction,
        neat.DefaultSpeciesSet,
        neat.DefaultStagnation,
        config_path
```

```
    )


    # tolide population
    population = neat.Population(config)

    # report mide dakhele consol
    population.add_reporter(neat.StdOutReporter(True))
    population.add_reporter(neat.StatisticsReporter())

    # 50 bar fitness k naghshe fitness functio
    # ra dare run mikone va me gen ha ro be fitness mide
    winner = population.run(fitness, 50)

# add kardane configuration
if __name__ == "__main__":
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, "config-feedforward.txt")
    run(config_path)
```

* باقی اطلاعات مورد نیاز این پروژه داخل کد به صورت کامنت نوشته شده است

توضیحاتی درمورد نحوه اجرای بازی:

ما 3 شئ در بازی در نظر گرفتیم : پرنده، لوله بالایی، لوله پایینی

پرتده : هر پرنده در جای خود فقط بالا و پایین میرود و میپرد پس دو تابع برای پریدن و حرکت آن در نظر گرفتیم

برای لوله ها ابتدا فاصله آن ها را برابر 500 قرار دادیم سپس موقعیت دو لوله را مشخص کردیم که به صورت عمودی چگونه مقابل هم قرار بگیرند (تابع set_height)

در تابع move به اندازه سرعتشان از موقعیت x آن ها کم کردیم (با این کار لوله ها به سمت راست شروع به حرکت میکنند)

در تابع collide بررسی میکنیم که پیکسل های پرنده با لوله ها تداخل دارد و یا نه، اگر داشت این تابع مقدار True را باز میگرداند که در مراحل بعدی دلیل آن بیان خواهد شد

*تابع draw در هر دو class ، bird و pipe آن ها را میکشد

تابع بعدی تابع draw_window است که در آن تمامی صفحه کشیده و آپدیت میشود

تابع بسیار مهم بعدی تابع fitness است که بزرگ ترین نقش را در برنامه بازی میکند :

ابتدا شبکه عصبی را create میکنیم سپس به ازای هر ژنوم یک پرنده با fitness برابر با 0 میسازیم و پرنده ها و ژن های آن ها را داخل دو آرایه birds و ge میریزیم یعنی اگر من پرنده خانه شماره 1 آرایه birds را بخواهم fitness آن هم برابر با خانه شماره 1 آرایه ge خواهد شد

کار مهم بعدی که در این تابع انجام میشود این است که چک میکند آیا پرنده ای باخته است یا نه؟

پرنده در 3 حالت میبازد:

۱. بالاتر از صفحه نمایش پرواز کند
۲. پایین تر از صفحه نمایش پرواز کند
۳. به لوله ها برخورد کند

هر کدام از این بخش در قسمت های مختلف این تابع بررسی میشوند که در کد به صورت خط به خط و واضح کامنت گزاشته شده

اگر پرنده ای باخته باشد، آن و fitness آن پرنده را از آرایه حذف میکنیم و در غیر این صورت به پرنده هایی که موفق شده اند تا کنون زنده بمانند fitness تعلق میگیرد

افزایش fitness پرنده در 2 مکان اتفاق میافتد :

۱. هر بار که پرنده میزانی جلو رفت 0.1 به fitness آن اضافه میشود و چون این کد در ثانیه 30 بار در حال اجرا است در هر ثانیه یکی به fitness پرنده زنده اضافه میشود
۲. هر بار پرنده از داخل لوله ها رد شود افزایش fitness به اندازه 2 خواهد داشت که باعث میشود پرنده بیشتر بخواهد از داخل لوله ها عبور کند

در تابع run با استفاده از config تولید جمعیت میکند ، در نهایت 50 بار این برنامه اجرا میشود و اگر تا 50 نسل پیشرفتی صورت نگیرد بازی تمام خواهد شد در غیر این صورت یعنی پرنده، برنده است و به امتیاز دلخواه ما رسیده که در این حالت برنامه تمام شده و در کنسول اعلام میشود که پرنده به هدفمان دستیابی پیدا کرده است.

پایان