In [1]:

```
import gym
import random
import numpy as np
import tflearn
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from statistics import median, mean
from collections import Counter

LR = 1e-3
env = gym.make("CartPole-v1")
env.reset()
goal_steps = 500
score_requirement = 50
initial_games = 10000
```

In [2]:

```
def some_random_games_first():

    for episode in range(5):
        env.reset()
        # each frame, up to 200
        for t in range(200):

            env.render()

            # action can be 0 or 1,left or right
            action = env.action_space.sample()
            observation, reward, done, info = env.step(action)
            if done:
                break
```

In [3]:

```
def initial_population():

    training_data = []
    scores = []
    accepted_scores = []

    for _ in range(initial_games):
        score = 0
        game_memory = []
        prev_observation = []
        # for each frame in 200
        for _ in range(goal_steps):
            # choose random action (0 or 1)
            action = random.randrange(0,2)
            observation, reward, done, info = env.step(action)

            if len(prev_observation) > 0 :
                game_memory.append([prev_observation, action])
            prev_observation = observation
            score = score + reward
            if done: break


        if score >= score_requirement:
            accepted_scores.append(score)
            for data in game_memory:
                # converting to 0 or 1 , For NN layer
                if data[1] == 1:
                    output = [0,1]
                elif data[1] == 0:
                    output = [1,0]

                # save training data
                training_data.append([data[0], output])


        env.reset()
        scores.append(score)

    training_data_save = np.array(training_data)
    np.save('saved.npy',training_data_save)

    print('Average accepted score:',mean(accepted_scores))
    print('Median score for accepted scores:',median(accepted_scores))
    print(Counter(accepted_scores))

    return training_data
```

In [4]:

```
def neural_network_model(input_size):

    network = input_data(shape=[None, input_size, 1], name='input')

    network = fully_connected(network, 128, activation='relu')
    network = dropout(network, 0.8)

    network = fully_connected(network, 256, activation='relu')
    network = dropout(network, 0.8)

    network = fully_connected(network, 512, activation='relu')
    network = dropout(network, 0.8)

    network = fully_connected(network, 256, activation='relu')
    network = dropout(network, 0.8)

    network = fully_connected(network, 128, activation='relu')
    network = dropout(network, 0.8)

    network = fully_connected(network, 2, activation='softmax')
    network = regression(network, optimizer='adam', learning_rate=LR, loss='cate
gorical_crossentropy', name='targets')
    model = tflearn.DNN(network, tensorboard_dir='log')

    return model
```

In [5]:

```
def train_model(training_data, model=False):

    X = np.array([i[0] for i in training_data]).reshape(-1,len(training_data[0][
0]),1)
    y = [i[1] for i in training_data]

    if not model:
        model = neural_network_model(input_size = len(X[0]))

    model.fit({'input': X}, {'targets': y}, n_epoch=5, snapshot_step=500, show_m
etric=True, run_id='openai_learning')
    return model
```

In [6]:

```
training_data = initial_population()
```

```
Average accepted score: 61.141689373297005
Median score for accepted scores: 58.0
Counter({51.0: 31, 50.0: 29, 53.0: 26, 54.0: 24, 52.0: 23, 56.0: 22,
58.0: 17, 55.0: 17, 67.0: 13, 60.0: 12, 61.0: 12, 59.0: 11, 63.0: 1
1, 70.0: 9, 57.0: 9, 64.0: 9, 62.0: 9, 68.0: 9, 73.0: 7, 65.0: 7, 6
9.0: 6, 79.0: 6, 71.0: 6, 66.0: 6, 80.0: 3, 81.0: 3, 72.0: 3, 85.0:
2, 76.0: 2, 75.0: 2, 82.0: 2, 99.0: 2, 88.0: 2, 78.0: 2, 128.0: 1, 8
6.0: 1, 96.0: 1, 114.0: 1, 74.0: 1, 122.0: 1, 117.0: 1, 84.0: 1, 92.
0: 1, 77.0: 1, 103.0: 1, 91.0: 1, 106.0: 1})
```

In [7]:

```
model = train_model(training_data)
```

```
Training Step: 1724  | total loss: 0.66223 | time: 4.631s
| Adam | epoch: 005 | loss: 0.66223 - acc: 0.6026 -- iter: 22016/220
72
Training Step: 1725  | total loss: 0.66317 | time: 4.642s
| Adam | epoch: 005 | loss: 0.66317 - acc: 0.6032 -- iter: 22072/220
72
--
```

In [8]:

```
scores = []
choices = []
for each_game in range(10):
    score = 0
    game_memory = []
    prev_obs = []
    env.reset()
    for _ in range(goal_steps):
        env.render()

        if len(prev_obs)==0:
            action = random.randrange(0,2)
        else:
            action = np.argmax(model.predict(prev_obs.reshape(-1,len(prev_obs),1
))[0])

        choices.append(action)

        new_observation, reward, done, info = env.step(action)
        prev_obs = new_observation
        game_memory.append([new_observation, action])
        score = score + reward
        if done: break

    scores.append(score)

print('Average Score:',sum(scores)/len(scores))
print('choice 1:{}  choice 0:{}'.format(choices.count(1)/len(choices),choices.co
unt(0)/len(choices)))
print(score_requirement)
```

```
Average Score: 374.1
choice 1:0.49799518845228546  choice 0:0.5020048115477145
50
```

In [ ]:

```

```