Author: Aditya Jain

About : Exercise 5, APS1080 (Introduction to RL)

Topic : REINFORCE

In [1]:
```
from comet_ml import Experiment

# Create an experiment with your api key
experiment = Experiment(
    api_key="epeaAhyRcHSkn92H4kusmbX8k",
    project_name="rl-uoft",
    workspace="adityajain07",
    log_code="True"
)
```

```
COMET WARNING: As you are running in a Jupyter environment, you will need to call `experiment.end()` when finished to en
sure all metrics and code are logged before exiting.
COMET ERROR: Failed to setup the std logger
COMET INFO: Experiment is live on comet.ml https://www.comet.ml/adityajain07/rl-uoft/4d4011434f9c4c63a4b2bd7924bec8fd
```

In [2]:
```
import matplotlib.pyplot as plt
import gym
from IPython import display as ipythondisplay
import numpy as np
import json
import pickle
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import torch
```

In [3]:
```
env = gym.make('CartPole-v0')

print('Observation Space: ', env.observation_space)
print('Action Space: ', env.action_space)

no_actions = 2
```

```
Observation Space:  Box([-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38], [4.8000002e+00 3.4028235e+38 4.18
87903e-01 3.4028235e+38], (4,), float32)
Action Space:  Discrete(2)
```

Initialize the Neural Network

In [5]:
```
input_dim  = 4              # cart position, cart velocity, pole angle, pole angular velocity
no_layers1 = 256
# no_layers2 = 128
no_layers3 = 128
out_dim    = no_actions     # no of actions

model = keras.Sequential()
model.add(keras.Input(shape=(input_dim,)))
model.add(layers.Dense(no_layers1, activation="relu"))
# model.add(layers.Dense(no_layers2, activation="relu"))
model.add(layers.Dense(no_layers3, activation="relu"))
model.add(layers.Dense(out_dim, activation="softmax"))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               1280
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 258
=================================================================
Total params: 34,434
Trainable params: 34,434
Non-trainable params: 0
_________________________________________________________________
```

Action Selection Function

In [6]:
```
def select_action(obs, no_actions, model):
    '''chooses action based on softmax over network's outputs'''

    obs      = tf.expand_dims(obs, axis=0)
    pred     = model(obs)
    pred    = pred.numpy()
    action  = np.random.choice(no_actions, p=pred[0])

    return action
```

REINFORCE Implementation

The network is trained for 1000 episodes

In [7]:
```
env         = gym.make('CartPole-v0')
converged   = False
episodes    = 0
alpha       = 1e-3
optimizer   = keras.optimizers.SGD(learning_rate=alpha)
gamma       = 0.9

while episodes<1000:
    episodes    += 1
    cur_obs      = env.reset()
    cur_action   = select_action(cur_obs, no_actions, model)
    done         = False

    trans_list   = []
    trans_list.append([cur_obs, cur_action])
    t_steps      = 0

    # interaction with env
    while not done:
        next_obs, reward, done, info = env.step(cur_action)
        t_steps                     += 1
        next_action                  = select_action(next_obs, no_actions, model)

        if done:
            trans_list.append([reward])
        else:
            trans_list.append([next_obs, next_action, reward])
            cur_obs     = next_obs
            cur_action = next_action


    # updating model parameters
    steps = len(trans_list)
    G     = 0
    for i in range(steps-2, -1, -1):
        G = trans_list[i+1][-1] + gamma*G

        with tf.GradientTape() as tape:
            prediction = model(tf.expand_dims(trans_list[i][0], axis=0), training=True)
            loss       = -tf.math.log(prediction[0][trans_list[i][1]]) * G

        grads = tape.gradient(loss, model.trainable_weights)
        optimizer.apply_gradients(zip(grads, model.trainable_weights))

    experiment.log_metric("Steps per episode", t_steps, step=episodes)
#     print('Episode no: ', episodes, ' - Steps survived: ', t_steps)

model.save('reinforce_e5.h5')
experiment.end()
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wil
l be empty until you train or evaluate the model.
COMET INFO: ---------------------------
COMET INFO: Comet.ml Experiment Summary
COMET INFO: ---------------------------
COMET INFO:   Data:
COMET INFO:     display_summary_level : 1
COMET INFO:     url                   : https://www.comet.ml/adityajain07/rl-uoft/4d4011434f9c4c63a4b2bd7924bec8fd
COMET INFO:   Metrics [count] (min, max):
COMET INFO:     Steps per episode [1000] : (9, 200)
COMET INFO:   Parameters:
COMET INFO:     Optimizer         : SGD
COMET INFO:     SGD_decay         : 0.0
COMET INFO:     SGD_learning_rate : 0.001
COMET INFO:     SGD_momentum      : 0.0
COMET INFO:     SGD_nesterov      : False
COMET INFO:   Uploads:
COMET INFO:     environment details      : 1
COMET INFO:     filename                 : 1
COMET INFO:     git metadata             : 1
COMET INFO:     git-patch (uncompressed) : 1 (59.83 KB)
COMET INFO:     installed packages       : 1
COMET INFO:     notebook                 : 1
COMET INFO:     source_code              : 1
COMET INFO: ---------------------------
COMET INFO: Uploading metrics, params, and assets to Comet before program termination (may take several seconds)
COMET INFO: The Python SDK has 3600 seconds to finish before aborting...
COMET INFO: Uploading 1 metrics, params and output messages
```

Check controller performance

In [11]:

```
def control_performance(env_name, model, no_actions, trials):
    env             = gym.make(env_name)
    steps_list      = []

    for i in range(trials):
        done            = False
        cur_obs         = env.reset()
        cur_action      = select_action(cur_obs, no_actions, model)
        t_steps         = 0

        while not done:
            next_obs, reward, done, info = env.step(cur_action)
            t_steps                      += 1
            next_action                  = select_action(next_obs, no_actions, model)

            cur_obs     = next_obs
            cur_action = next_action

        steps_list.append(t_steps)

    return steps_list
```

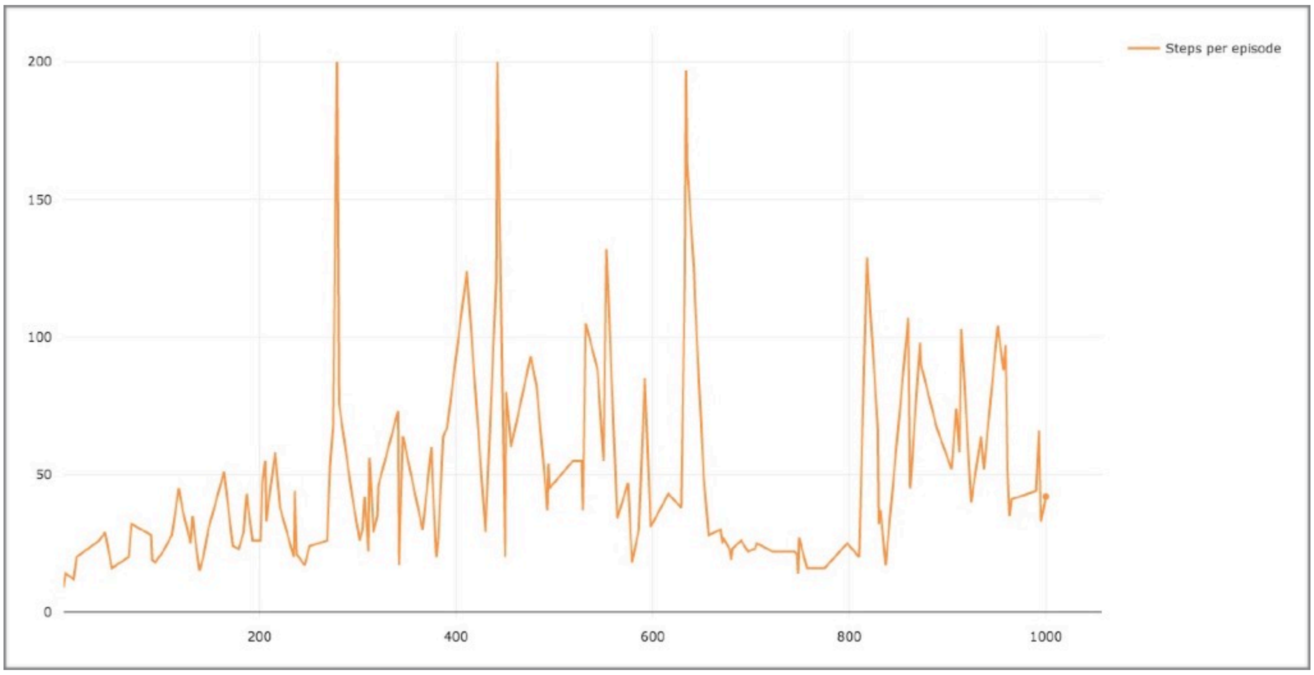
In [12]:

```
steps = control_performance('CartPole-v0', model, no_actions, 100)
print('Average steps the pole is sustained in 100 trials: ', sum(steps)/len(steps))
```

```
Average steps the pole is sustained in 100 trials:  84.0
```

In [ ]:

The below plot shows the number of steps the pole was sustained in each episode during training (for 1000 episodes).


Steps per episode