# Deep Reinforcement Algorithm in OpenAI gym environment

We shall build a deep neural network and use RL to solve a cart and pole balancing problem

In [1]:
```
import sys
print(sys.version)
```

```
3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
```

In git bash, we type the following commands:

git clone https://github.com/openai/gym (https://github.com/openai/gym)

cd gym

pip install -e . # minimal install

This downloads the bare minimums for the OpenAI Gym environment.

In [2]:
```
import gym
print(gym.__version__)

import keras
print(keras.__version__)
```

```
0.12.1

Using Theano backend.
WARNING (theano.configdefaults): g++ not available, if using conda: `conda install m2w64-toolchain`
C:\ProgramData\Anaconda3\lib\site-packages\theano\configdefaults.py:560: UserWarning: DeprecationWarning: there is no
c++ compiler.This is deprecated and with Theano 0.11 a c++ compiler will be mandatory
  warnings.warn("DeprecationWarning: there is no c++ compiler."
WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute optimized C-implementations (for
both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this w
arning, set Theano flags cxx to an empty string.
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.

2.2.4
```

If it does not show 'Using Theano backend' and instead shows "Using Tensorflow backend" or anything else; go to .keras folder in the directory where Anaconda is installed; open the 'keras' JSON file in a text editor and change whatever is written in the section marked as "backend" to "Theano"

In [3]:
```
import random
import math
import numpy as np
from collections import deque
```

## Setting up OpenAI Gym environment

In [4]:

```
env = gym.make('CartPole-v0')

for i_episode in range(20):
    observation = env.reset()

    for t in range(100):
        env.render()

        print(observation)

        action = env.action_space.sample()

        observation, reward, done, info = env.step(action)

        if done:
            break
```

```
[ 0.02093232  0.04103462 -0.00739778 -0.03682714]
[ 0.02175302  0.23626187 -0.00813433 -0.33183494]
[ 0.02647825  0.43149866 -0.01477102 -0.62707189]
[ 0.03510823  0.62682363 -0.02731246 -0.92436991]
[ 0.0476447   0.43208105 -0.04579986 -0.64039385]
[ 0.05628632  0.23762652 -0.05860774 -0.36247838]
[ 0.06103885  0.04338442 -0.06585731 -0.0888363 ]
[ 0.06190654 -0.15073476 -0.06763403  0.1823632 ]
[ 0.05889184 -0.34482702 -0.06398677  0.45296671]
[ 0.0519953  -0.14886134 -0.05492743  0.14082046]
[ 0.04901808  0.04700251 -0.05211102 -0.16867278]
[ 0.04995813 -0.14733631 -0.05548448  0.10712603]
[ 0.0470114  -0.34162106 -0.05334196  0.38180063]
[ 0.04017898 -0.14578388 -0.04570595  0.07278757]
[ 0.0372633   0.04996251 -0.04425019 -0.23395825]
[ 0.03826255  0.24568786 -0.04892936 -0.5402642 ]
[ 0.04317631  0.44146222 -0.05973464 -0.84795377]
[ 0.05200555  0.24720369 -0.07669372 -0.57463724]
[ 0.05694963  0.44331232 -0.08818646 -0.89046133]
[ 0.06581587  0.63951296 -0.10599569 -1.20951189]
[ 0.07860613  0.44590734 -0.13018593 -0.95183774]
[ 0.08752428  0.25275486 -0.14922268 -0.70272741]
[ 0.09257938  0.05998149 -0.16327723 -0.46048964]
[ 0.09377901 -0.1325017  -0.17248702 -0.22339543]
[ 0.09112897  0.06461197 -0.17695493 -0.56513857]
[ 0.09242121  0.2617173  -0.1882577  -0.90793333]
[ 0.09765556  0.06957381 -0.20641637 -0.67983174]
[-0.04523021  0.04879035  0.00477001 -0.00169103]
[-0.0442544  -0.14639969  0.00473619  0.29249307]
[-0.04718239 -0.34158885  0.01058605  0.58666596]
[-0.05401417 -0.53685745  0.02231937  0.88266469]
[-0.06475132 -0.34204562  0.03997267  0.59708108]
[-0.07159223 -0.53770347  0.05191429  0.90208225]
[-0.0823463  -0.73348884  0.06995593  1.21062059]
[-0.09701608 -0.92944079  0.09416835  1.52437987]
[-0.11560489 -0.73557363  0.12465594  1.26251214]
[-0.13031637 -0.5422463   0.14990619  1.01132533]
[-0.14116129 -0.34940785  0.17013269  0.7692192 ]
[-0.14814945 -0.15698492  0.18551708  0.53453008]
[-0.15128915 -0.35416453  0.19620768  0.87945409]
[-0.0324465   0.03512596 -0.01331526 -0.01065901]
```

```
[-0.03174398  0.23043631 -0.01352844 -0.30751314]
[-0.02713526  0.03550972 -0.01967871 -0.01912724]
[-0.02642506  0.23090827 -0.02006125 -0.3179535 ]
[-0.0218069   0.42631012 -0.02642032 -0.6168949 ]
[-0.01328069  0.23156704 -0.03875822 -0.33264889]
[-0.00864935  0.03701757 -0.0454112  -0.05243592]
[-0.007909    0.23276025 -0.04645992 -0.35909356]
[-0.0032538   0.42851081 -0.05364179 -0.66605685]
[ 0.00531642  0.23417436 -0.06696292 -0.39073459]
[ 0.00999991  0.43017961 -0.07477762 -0.70375593]
[ 0.0186035   0.23616922 -0.08885274 -0.43551774]
[ 0.02332688  0.43242925 -0.09756309 -0.75483483]
[ 0.03197547  0.23877794 -0.11265979 -0.49437728]
[ 0.03675103  0.43529342 -0.12254733 -0.82033377]
[ 0.0454569   0.63186016 -0.13895401 -1.14891145]
[ 0.0580941   0.43879809 -0.16193224 -0.90283076]
[ 0.06687006  0.24619622 -0.17998885 -0.66510777]
[ 0.07179399  0.05397367 -0.19329101 -0.43406326]
[ 0.07287346  0.25123128 -0.20197227 -0.78091386]
[ 0.02039814 -0.02666361  0.01630458 -0.00255631]
[ 0.01986487 -0.22201555  0.01625345  0.29522598]
[ 0.01542456 -0.4173654   0.02215797  0.59299036]
[ 0.00707725 -0.22256052  0.03401778  0.30736868]
[ 0.00262604 -0.02793939  0.04016515  0.02560519]
[ 0.00206725 -0.22361364  0.04067725  0.33068522]
[-0.00240502 -0.4192903   0.04729096  0.63591323]
[-0.01079083 -0.61503881  0.06000922  0.94310607]
[-0.0230916  -0.4207745   0.07887135  0.66986625]
[-0.03150709 -0.22683259  0.09226867  0.4030219 ]
[-0.03604375 -0.03313213  0.10032911  0.14079565]
[-0.03670639  0.16042045  0.10314502 -0.11862593]
[-0.03349798  0.35392493  0.1007725  -0.37706912]
[-0.02641948  0.15752691  0.09323112 -0.05438966]
[-0.02326894 -0.0387996   0.09214333  0.26619144]
[-0.02404494  0.15489471  0.09746716  0.00393431]
[-0.02094704  0.34849353  0.09754584 -0.25647564]
[-0.01397717  0.54209728  0.09241633 -0.5168677 ]
[-0.00313522  0.73580472  0.08207898 -0.7790572 ]
[ 0.01158087  0.92970799  0.06649783 -1.04482879]
[ 0.03017503  0.73376925  0.04560126 -0.7320343 ]
[ 0.04485041  0.53804781  0.03096057 -0.42535521]
[ 0.05561137  0.3425013   0.02245347 -0.12307506]
```

```
[0.0624614  0.14706498 0.01999196 0.17660629]
[ 0.0654027   0.3418952   0.02352409 -0.1097034 ]
[0.0722406  0.14644419 0.02133002 0.19030744]
[ 0.07516948  0.3412546   0.02513617 -0.09557115]
[ 0.08199458  0.53600744  0.02322475 -0.38021887]
[ 0.09271472  0.34056352  0.01562037 -0.08030459]
[0.099526   0.14522117 0.01401428 0.21726539]
[ 0.10243042 -0.05009829  0.01835959  0.51433587]
[0.10142845 0.14476035 0.0286463  0.22749459]
[ 0.10432366 -0.05075902  0.0331962   0.52907418]
[ 0.10330848 -0.24633188  0.04377768  0.83202973]
[ 0.09838184 -0.44202388  0.06041827  1.13815303]
[ 0.08954136 -0.24774183  0.08318133  0.86501386]
[ 0.08458653 -0.4438915   0.10048161  1.18264818]
[ 0.0757087  -0.25020665  0.12413457  0.92307824]
[ 0.07070456 -0.05696058  0.14259614  0.67184153]
[ 0.06956535 -0.25374657  0.15603297  1.0058059 ]
[ 0.06449042 -0.0610134   0.17614909  0.76590605]
[0.06327015 0.1313024  0.19146721 0.53341666]
[ 0.0658962  -0.06592342  0.20213554  0.87979731]
[-0.03030504 -0.01952069 -0.01259421  0.04597901]
[-0.03069545  0.17577957 -0.01167463 -0.25065074]
[-0.02717986 -0.01917374 -0.01668764  0.03832702]
[-0.02756333 -0.21405246 -0.0159211   0.32569847]
[-0.03184438 -0.40894416 -0.00940713  0.61331835]
[-0.04002327 -0.60393339  0.00285923  0.90302361]
[-0.05210193 -0.79909396  0.0209197   1.19660387]
[-0.06808381 -0.99448036  0.04485178  1.49576929]
[-0.08797342 -0.79993154  0.07476717  1.21742154]
[-0.10397205 -0.6058491   0.0991156   0.94907228]
[-0.11608903 -0.41219111  0.11809704  0.68910329]
[-0.12433285 -0.2188889   0.13187911  0.43580949]
[-0.12871063 -0.0258563   0.1405953   0.18743504]
[-0.12922776 -0.22268052  0.144344    0.52095716]
[-0.13368137 -0.02985388  0.15476314  0.27701509]
[-0.13427845 -0.22680613  0.16030345  0.61423147]
[-0.13881457 -0.03424421  0.17258807  0.3760195 ]
[-0.13949945  0.15806014  0.18010846  0.14233776]
[-0.13633825  0.35020695  0.18295522 -0.0885542 ]
[-0.12933411  0.54229935  0.18118414 -0.31839461]
[-0.11848812  0.34512174  0.17481624  0.02550758]
[-0.11158569  0.14797991  0.17532639  0.36784781]
```

```
[-0.10862609  0.34023366  0.18268335  0.13517045]
[-0.10182142  0.14302913  0.18538676  0.47946561]
[-0.09896084 -0.05415947  0.19497607  0.82437233]
[-0.02234447 -0.01165295  0.01263744 -0.01553765]
[-0.02257753 -0.20695384  0.01232669  0.28110559]
[-0.02671661 -0.01200987  0.0179488  -0.00766419]
[-0.02695681  0.18285013  0.01779552 -0.29463045]
[-0.0232998  -0.01252094  0.01190291  0.00361137]
[-0.02355022  0.1824283   0.01197513 -0.28529241]
[-0.01990166 -0.01286238  0.00626929  0.01114321]
[-0.0201589   0.1821691   0.00649215 -0.27955511]
[-0.01651552  0.37719785  0.00090105 -0.57018337]
[-0.00897156  0.57230715 -0.01050262 -0.8625823 ]
[ 0.00247458  0.37732976 -0.02775427 -0.57322006]
[ 0.01002117  0.57282963 -0.03921867 -0.87451564]
[ 0.02147777  0.7684622  -0.05670898 -1.17926617]
[ 0.03684701  0.57412058 -0.0802943  -0.90488586]
[ 0.04832942  0.38017244 -0.09839202 -0.63848214]
[ 0.05593287  0.57651869 -0.11116166 -0.96045763]
[ 0.06746324  0.38305222 -0.13037082 -0.70466334]
[ 0.07512429  0.18995468 -0.14446408 -0.45569557]
[ 0.07892338  0.38679208 -0.15357799 -0.79020092]
[ 0.08665922  0.19407499 -0.16938201 -0.54950233]
[ 0.09054072  0.39112056 -0.18037206 -0.89040137]
[ 0.09836314  0.19884369 -0.19818009 -0.65940733]
[ 0.02640999 -0.03542804  0.02225571  0.02266707]
[ 0.02570143 -0.23086197  0.02270905  0.32228801]
[ 0.02108419 -0.03607063  0.02915481  0.03685223]
[ 0.02036278 -0.23159827  0.02989185  0.33858935]
[ 0.01573081 -0.42713254  0.03666364  0.64054662]
[ 0.00718816 -0.23254039  0.04947457  0.35963125]
[ 0.00253735 -0.42832944  0.0566672   0.66749506]
[-0.00602924 -0.62419175  0.0700171   0.97746819]
[-0.01851307 -0.43007498  0.08956646  0.70757467]
[-0.02711457 -0.62631601  0.10371796  1.02705365]
[-0.03964089 -0.82265425  0.12425903  1.35041679]
[-0.05609398 -0.62929289  0.15126736  1.09904927]
[-0.06867983 -0.8260468   0.17324835  1.43511319]
[-0.08520077 -1.02282972  0.20195061  1.77654941]
[ 0.00267365  0.00435632 -0.02337877  0.04709425]
[ 0.00276078 -0.19042273 -0.02243689  0.33231025]
[-0.00104768 -0.38521826 -0.01579068  0.6178341 ]
```

```
[-0.00875204 -0.58011611 -0.003434    0.90550216]
[-0.02035437 -0.77519139  0.01467604  1.19710377]
[-0.03585819 -0.58026244  0.03861812  0.90905643]
[-0.04746344 -0.38568387  0.05679925  0.6287571 ]
[-0.05517712 -0.19139868  0.06937439  0.35448927]
[-0.05900509 -0.38743494  0.07646417  0.66821605]
[-0.06675379 -0.19345488  0.08982849  0.40055473]
[-0.07062289  0.00028568  0.09783959  0.13748967]
[-0.07061718 -0.19609159  0.10058938  0.45936578]
[-0.07453901 -0.00252468  0.1097767   0.20000652]
[-0.0745895   0.19086996  0.11377683 -0.05612969]
[-0.0707721   0.38419226  0.11265423 -0.31085994]
[-0.06308826  0.18766064  0.10643704  0.01511916]
[-0.05933504  0.3811079   0.10673942 -0.2421762 ]
[-0.05171289  0.57455609  0.10189589 -0.49937521]
[-0.04022176  0.37815634  0.09190839 -0.17639806]
[-0.03265864  0.18184748  0.08838043  0.14380546]
[-0.02902169  0.37559985  0.09125654 -0.11973873]
[-0.02150969  0.56930387  0.08886176 -0.38229357]
[-0.01012361  0.76305911  0.08121589 -0.64568871]
[ 0.00513757  0.95696104  0.06830212 -0.91173167]
[ 0.02427679  0.76098472  0.05006748 -0.59838741]
[ 0.03949648  0.56519929  0.03809974 -0.29036359]
[ 0.05080047  0.75975785  0.03229246 -0.57079102]
[ 0.06599563  0.56419827  0.02087664 -0.26811226]
[ 0.07727959  0.75901616  0.0155144  -0.55413813]
[ 0.09245992  0.95391687  0.00443164 -0.84189295]
[ 0.11153825  1.14897805 -0.01240622 -1.13317895]
[ 0.13451781  0.95402065 -0.0350698  -0.8444127 ]
[ 0.15359823  1.14960316 -0.05195806 -1.14791448]
[ 0.17659029  0.95519663 -0.07491635 -0.871967  ]
[ 0.19569422  0.76116908 -0.09235569 -0.60374684]
[ 0.2109176   0.95745308 -0.10443062 -0.92403205]
[ 0.23006667  1.15381889 -0.12291126 -1.24762292]
[ 0.25314304  0.96046844 -0.14786372 -0.99583021]
[ 0.27235241  0.76760006 -0.16778033 -0.7529982 ]
[ 0.28770441  0.57513977 -0.18284029 -0.5174579 ]
[ 0.29920721  0.38299925 -0.19318945 -0.28750728]
[ 0.30686719  0.58027584 -0.19893959 -0.63436228]
[ 0.00368904 -0.02360693 -0.00412563 -0.03157786]
[ 0.0032169  -0.21866948 -0.00475718  0.25980054]
[-0.00115649 -0.02347994  0.00043883 -0.03437906]
```

```
[-1.62609188e-03 -2.18608182e-01 -2.48753829e-04  2.58442289e-01]
[-0.00599826 -0.02348268  0.00492009 -0.03431909]
[-0.00646791 -0.21867484  0.00423371  0.25991211]
[-0.01084141 -0.41385697  0.00943195  0.55392739]
[-0.01911855 -0.60911009  0.0205105   0.84956699]
[-0.03130075 -0.41427376  0.03750184  0.56340364]
[-0.03958622 -0.21969754  0.04876991  0.28276747]
[-0.04398017 -0.41547997  0.05442526  0.5904246 ]
[-0.05228977 -0.61132001  0.06623375  0.89974283]
[-0.06451617 -0.41715513  0.08422861  0.62859261]
[-0.07285928 -0.2233034   0.09680046  0.36357979]
[-0.07732534 -0.41965828  0.10407206  0.68514817]
[-0.08571851 -0.61605936  0.11777502  1.0086978 ]
[-0.0980397  -0.81253966  0.13794898  1.33592245]
[-0.11429049 -0.6193989   0.16466743  1.08939202]
[-0.12667847 -0.42678498  0.18645527  0.85257313]
[-0.13521417 -0.23462724  0.20350673  0.62383315]
[ 0.00970341 -0.03852314 -0.04587602 -0.02002788]
[ 0.00893295 -0.23295818 -0.04627658  0.25783501]
[ 0.00427379 -0.42738996 -0.04111988  0.53556999]
[-0.00427401 -0.23171465 -0.03040848  0.2302193 ]
[-0.00890831 -0.03617167 -0.02580409 -0.07189829]
[-0.00963174 -0.23091436 -0.02724206  0.21253294]
[-0.01425003 -0.42563644 -0.0229914   0.49649951]
[-0.02276276 -0.62042677 -0.01306141  0.78184891]
[-0.03517129 -0.81536678  0.00257557  1.07039404]
[-0.05147863 -1.0105227   0.02398345  1.36388416]
[-0.07168908 -0.81570925  0.05126113  1.07879848]
[-0.08800327 -0.62130034  0.0728371   0.80263234]
[-0.10042927 -0.81734149  0.08888975  1.11730955]
[-0.1167761  -0.6234913   0.11123594  0.85378147]
[-0.12924593 -0.43004698  0.12831157  0.59804428]
[-0.13784687 -0.23693179  0.14027246  0.34837297]
[-0.1425855  -0.43374112  0.14723992  0.68179234]
[-0.15126033 -0.24093752  0.16087576  0.43884851]
[-0.15607908 -0.04841452  0.16965273  0.20088669]
[-0.15704737  0.14392623  0.17367047 -0.03384343]
[-0.15416884 -0.05320574  0.1729936   0.30820921]
[-0.15523296  0.13908351  0.17915778  0.07468736]
[-0.15245129  0.33124517  0.18065153 -0.15655049]
[-0.14582638  0.52338277  0.17752052 -0.38724372]
[-0.13535873  0.71559921  0.16977564 -0.6191199 ]
```

```
[-0.12104674  0.90799415  0.15739325 -0.85388852]
[-0.10288686  1.10066108  0.14031548 -1.09323386]
[-0.08087364  1.29368383  0.1184508  -1.33880491]
[-0.05499996  1.48713185  0.0916747  -1.59220051]
[-0.02525733  1.29104916  0.05983069 -1.27239694]
[ 5.63657660e-04  1.48535871e+00  3.43827522e-02 -1.54576001e+00]
[ 0.03027083  1.68005125  0.00346755 -1.82751968]
[ 0.06387186  1.48489104 -0.03308284 -1.53376166]
[ 0.09356968  1.29018288 -0.06375807 -1.25158381]
[ 0.11937334  1.09593316 -0.08878975 -0.97953344]
[ 0.141292    1.29212592 -0.10838042 -1.29873322]
[ 0.16713452  1.48844399 -0.13435508 -1.623284  ]
[ 0.1969034   1.29513522 -0.16682076 -1.37531568]
[ 0.2228061   1.10244391 -0.19432708 -1.13910937]
[ 0.01452846 -0.00299311  0.02563147  0.04723443]
[ 0.0144686   0.1917521   0.02657615 -0.23725261]
[ 0.01830364  0.38648449  0.0218311  -0.52143558]
[ 0.02603333  0.19106215  0.01140239 -0.22195409]
[ 0.02985458  0.38601928  0.00696331 -0.51101854]
[ 0.03757496  0.58104246 -0.00325706 -0.80149902]
[ 0.04919581  0.38596533 -0.01928704 -0.50984245]
[ 0.05691512  0.19112032 -0.02948389 -0.22329934]
[ 0.06073752 -0.00356808 -0.03394988  0.05993934]
[ 0.06066616 -0.19818723 -0.03275109  0.34172049]
[ 0.05670242 -0.39282828 -0.02591668  0.62389839]
[ 0.04884585 -0.58757899 -0.01343871  0.9083077 ]
[ 0.03709427 -0.39227772  0.00472744  0.6114314 ]
[ 0.02924872 -0.19722216  0.01695607  0.32024119]
[ 0.02530428 -0.39258144  0.02336089  0.61822282]
[ 0.01745265 -0.19779346  0.03572535  0.33298806]
[ 0.01349678 -0.39340521  0.04238511  0.63671947]
[ 0.00562867 -0.58909183  0.0551195   0.94244327]
[-0.00615316 -0.78491145  0.07396836  1.25192354]
[-0.02185139 -0.59081101  0.09900683  0.98329588]
[-0.03366761 -0.39714491  0.11867275  0.72327984]
[-0.04161051 -0.59369086  0.13313835  1.05083315]
[-0.05348433 -0.79030321  0.15415501  1.38216884]
[-0.06929039 -0.5974038   0.18179839  1.14139321]
[-0.08123847 -0.79437502  0.20462625  1.48513575]
[-0.0076605   0.0310948   0.04732376  0.00859401]
[-0.00703861 -0.16467277  0.04749564  0.3158246 ]
[-0.01033206  0.02974161  0.05381213  0.03849037]
```

```
[-0.00973723 -0.16610907  0.05458194  0.34765426]
[-0.01305941  0.02819577  0.06153502  0.07267017]
[-0.0124955  -0.16775192  0.06298843  0.38411529]
[-0.01585053 -0.36370895  0.07067073  0.69597429]
[-0.02312471 -0.55973617  0.08459022  1.01004105]
[-0.03431944 -0.75587895  0.10479104  1.3280431 ]
[-0.04943702 -0.56222392  0.1313519   1.06990623]
[-0.06068149 -0.36906035  0.15275003  0.82116332]
[-0.0680627  -0.56590548  0.16917329  1.15772711]
[-0.07938081 -0.76277908  0.19232784  1.49832178]
[-0.00238972  0.01134746 -0.00916745  0.04244146]
[-0.00216277 -0.18364184 -0.00831862  0.33221793]
[-0.00583561  0.01159752 -0.00167426  0.03692337]
[-0.00560366 -0.18350038 -0.00093579  0.32907758]
[-0.00927366 -0.378609    0.00564576  0.62146525]
[-0.01684584 -0.18356634  0.01807506  0.33056578]
[-0.02051717 -0.37894086  0.02468638  0.62889347]
[-0.02809599 -0.18417197  0.03726425  0.34408619]
[-0.03177943 -0.37980367  0.04414597  0.64828319]
[-0.0393755  -0.57551194  0.05711164  0.95453427]
[-0.05088574 -0.38120285  0.07620232  0.68032756]
[-0.0585098  -0.18721742  0.08980887  0.41257562]
[-0.06225414  0.00652432  0.09806039  0.14950255]
[-0.06212366  0.2001153   0.10105044 -0.11070531]
[-0.05812135  0.39365498  0.09883633 -0.36987559]
[-0.05024825  0.58724393  0.09143882 -0.62983126]
[-0.03850337  0.78097896  0.07884219 -0.89237421]
[-0.02288379  0.97494803  0.06099471 -1.15926838]
[-0.00338483  0.77908666  0.03780934 -0.84810135]
[ 0.0121969   0.97367307  0.02084732 -1.12865906]
[ 0.03167036  1.16851584 -0.00172587 -1.41473104]
[ 0.05504068  1.36365913 -0.03002049 -1.70795294]
[ 0.08231386  1.55911305 -0.06417955 -2.00982642]
[ 0.11349612  1.36471476 -0.10437607 -1.73768516]
[ 0.14079042  1.56086006 -0.13912978 -2.06093278]
[ 0.17200762  1.36740535 -0.18034843 -1.81432704]
[-0.00307058 -0.01863247  0.01663952 -0.02440232]
[-0.00344323 -0.21398904  0.01615147  0.27348377]
[-0.00772301 -0.40933768  0.02162115  0.57121684]
[-0.01590977 -0.21452549  0.03304548  0.28542298]
[-0.02020028 -0.41010277  0.03875394  0.58834234]
[-0.02840233 -0.21554434  0.05052079  0.30811462]
```

```
[-0.03271322 -0.41134839  0.05668308  0.61629274]
[-0.04094019 -0.21706225  0.06900894  0.34198749]
[-0.04528143 -0.41309468  0.07584869  0.65560878]
[-0.05354332 -0.60918608  0.08896086  0.97117834]
[-0.06572705 -0.80538208  0.10838443  1.29042858]
[-0.08183469 -0.61179239  0.134193    1.03355065]
[-0.09407054 -0.80841888  0.15486402  1.3651732 ]
[-0.11023891 -0.61553788  0.18216748  1.12466241]
[-0.12254967 -0.81251838  0.20466073  1.46850561]
[-0.01812699  0.01898009  0.00548304  0.0208221 ]
[-0.01774738 -0.17622006  0.00589948  0.31522993]
[-0.02127179  0.01881736  0.01220408  0.02441333]
[-0.02089544 -0.17647746  0.01269234  0.32092166]
[-0.02442499  0.01846146  0.01911078  0.03226828]
[-0.02405576  0.21330422  0.01975614 -0.25432426]
[-0.01978967  0.01790583  0.01466966  0.04452404]
[-0.01943156 -0.17742337  0.01556014  0.34179904]
[-0.02298003 -0.37276321  0.02239612  0.63934782]
[-0.03043529 -0.56819014  0.03518307  0.93899862]
[-0.04179909 -0.37355971  0.05396305  0.65757534]
[-0.04927029 -0.17922878  0.06711455  0.3823607 ]
[-0.05285486 -0.3752363   0.07476177  0.69542692]
[-0.06035959 -0.57131117  0.08867031  1.01067716]
[-0.07178581 -0.76749731  0.10888385  1.32983517]
[-0.08713576 -0.96381138  0.13548055  1.65451106]
[-0.10641198 -1.16022973  0.16857077  1.98614832]
[-0.12961658 -0.9672316   0.20829374  1.75008002]
[-0.02588376  0.01080375  0.0391173  -0.04713182]
[-0.02566768 -0.18485665  0.03817466  0.25763181]
[-0.02936481  0.00970009  0.0433273  -0.02277005]
[-0.02917081  0.20417477  0.0428719  -0.30147406]
[-0.02508732  0.39866029  0.03684241 -0.5803338 ]
[-0.01711411  0.20304196  0.02523574 -0.27627622]
[-0.01305327  0.00756923  0.01971021  0.02425803]
[-0.01290189 -0.18782975  0.02019537  0.32309401]
[-0.01665848 -0.38323337  0.02665725  0.6220767 ]
[-0.02432315 -0.57871722  0.03909879  0.9230346 ]
[-0.03589749 -0.38414469  0.05755948  0.64289098]
[-0.04358039 -0.58001968  0.0704173   0.95313003]
[-0.05518078 -0.38591223  0.0894799   0.68337602]
[-0.06289903 -0.58215524  0.10314742  1.00283488]
[-0.07454213 -0.77849268  0.12320412  1.32604733]
```

```
[-0.09011198 -0.97493582  0.14972507  1.65460893]
[-0.1096107  -1.17145473  0.18281724  1.98994424]
[0.02113885 0.04086786 0.02803656 0.04156319]
[ 0.0219562  -0.15464468  0.02886782  0.34295841]
[ 0.01886331 -0.35016518  0.03572699  0.64460292]
[ 0.01186001 -0.54576633  0.04861905  0.94831891]
[ 0.00094468 -0.35133151  0.06758542  0.67129969]
[-0.00608195 -0.15721098  0.08101142  0.40063865]
[-0.00922617  0.036674    0.08902419  0.13455631]
[-0.00849269  0.23041547  0.09171532 -0.1287665 ]
[-0.00388438  0.03410752  0.08913999  0.19138435]
[-0.00320223 -0.16216908  0.09296767  0.51080277]
[-0.00644561 -0.35846923  0.10318373  0.8312746 ]
[-0.013615   -0.55483863  0.11980922  1.15454426]
[-0.02471177 -0.36146522  0.14290011  0.90170316]
[-0.03194107 -0.16853822  0.16093417  0.65713217]
[-0.03531184 -0.36549102  0.17407681  0.9958545 ]
[-0.04262166 -0.56245944  0.1939939   1.33776241]
[-0.04891654 -0.04602267  0.049981   -0.02537302]
[-0.04983699 -0.24182443  0.04947354  0.28265115]
[-0.05467348 -0.04744178  0.05512656  0.00597321]
[-0.05562232 -0.24330921  0.05524602  0.31552706]
[-0.0604885  -0.04901594  0.06155657  0.04076587]
[-0.06146882 -0.24496412  0.06237188  0.35221764]
[-0.0663681  -0.440915    0.06941624  0.66389788]
[-0.0751864  -0.63693046  0.08269419  0.97760495]
[-0.08792501 -0.83305804  0.10224629  1.2950754 ]
[-0.10458617 -1.02931953  0.1281478   1.6179382 ]
[-0.12517256 -0.8359201   0.16050656  1.36779019]
[-0.14189096 -1.03264525  0.18786237  1.70607157]
[-0.02680926  0.03664241 -0.04592379  0.02637377]
[-0.02607641 -0.1577919  -0.04539631  0.30422089]
[-0.02923225  0.03794661 -0.03931189 -0.00242632]
[-0.02847332 -0.15659014 -0.03936042  0.27759864]
[-0.03160512  0.03907058 -0.03380845 -0.02723412]
[-0.03082371 -0.15555063 -0.03435313  0.25459309]
[-0.03393472 -0.35016567 -0.02926127  0.53624557]
[-0.04093803 -0.15464476 -0.01853636  0.23448817]
[-0.04403093 -0.34949703 -0.01384659  0.52126699]
[-0.05102087 -0.15418293 -0.00342125  0.22425316]
[-0.05410453  0.04098775  0.00106381 -0.06950699]
[-5.32847727e-02  2.36094434e-01 -3.26330147e-04 -3.61854093e-01]
```

```
[-0.04856288  0.43122102 -0.00756341 -0.6546399 ]
[-0.03993846  0.62644746 -0.02065621 -0.9496948 ]
[-0.02740951  0.43160957 -0.03965011 -0.66357281]
[-0.01877732  0.237061   -0.05292156 -0.3836334 ]
[-0.0140361   0.43289282 -0.06059423 -0.69252155]
[-0.00537825  0.62880076 -0.07444466 -1.00364794]
[ 0.00719777  0.82483397 -0.09451762 -1.31874985]
[ 0.02369445  1.0210154  -0.12089262 -1.63945534]
[ 0.04411476  1.21732913 -0.15368172 -1.96723263]
[ 0.06846134  1.02412996 -0.19302638 -1.72585416]
[0.00640668 0.01631727 0.03540607 0.00128093]
[ 0.00673302 -0.1792941   0.03543169  0.3049213 ]
[ 0.00314714 -0.37490259  0.04153012  0.60856462]
[-0.00435091 -0.18038509  0.05370141  0.3292463 ]
[-0.00795861 -0.37622876  0.06028633  0.63836894]
[-0.01548319 -0.18199694  0.07305371  0.36526395]
[-0.01912313  0.01201491  0.08035899  0.09648108]
[-0.01888283  0.20589867  0.08228861 -0.16980593]
[-0.01476486  0.39975228  0.07889249 -0.43543684]
[-0.00676981  0.2036073   0.07018376 -0.11896417]
[-0.00269766  0.00755363  0.06780447  0.19500944]
[-0.00254659 -0.1884694   0.07170466  0.50828759]
[-0.00631598  0.0055729   0.08187041  0.23903518]
[-0.00620452  0.19943564  0.08665112 -0.02674189]
[-0.00221581  0.00318481  0.08611628  0.29197346]
[-0.00215211  0.19698008  0.09195575  0.02764469]
[ 0.00178749  0.39067123  0.09250864 -0.23466764]
[ 0.00960091  0.58435806  0.08781529 -0.49679684]
[ 0.02128807  0.77813913  0.07787935 -0.76056388]
[ 0.03685086  0.58203554  0.06266808 -0.44442672]
[ 0.04849157  0.3860855   0.05377954 -0.13266626]
[ 0.05621328  0.5803975   0.05112622 -0.4079093 ]
[ 0.06782123  0.38458931  0.04296803 -0.09955625]
[ 0.07551301  0.57906996  0.04097691 -0.37837924]
[ 0.08709441  0.38339074  0.03340932 -0.07306298]
[0.09476223 0.18780615 0.03194806 0.22997071]
[ 0.09851835 -0.00775742  0.03654748  0.53255751]
[0.0983632  0.18683195 0.04719863 0.25161093]
[ 0.10209984 -0.00893109  0.05223084  0.55879972]
[ 0.10192122 -0.20474579  0.06340684  0.8674701 ]
[ 0.0978263  -0.40067058  0.08075624  1.17939582]
[ 0.08981289 -0.20668465  0.10434416  0.91308178]
```

```
[ 0.0856792  -0.01311724  0.12260579  0.6549314 ]
[ 0.08541685 -0.20971364  0.13570442  0.98356818]
[ 0.08122258 -0.01664496  0.15537578  0.73640244]
[ 0.08088968 -0.21353266  0.17010383  1.07367317]
[ 0.07661903 -0.41044407  0.1915773   1.41454701]
```

In [5]:
```
print(env.action_space)
print(env.observation_space)
```

```
Discrete(2)
Box(4,)
```

In [6]:
```
print(env.observation_space.high)
print(env.observation_space.low)
```

```
[4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]
[-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38]
```

## Defining parameters

In [7]:

```
# training parameters

n_episodes = 1000    # no. of episodes
n_win_ticks = 195    # every time step is a tick(in OpenAI); done state = win_tick
max_env_steps = None # for OPen AI

# RL parameters

gamma = 1.0          # Discount factor: measure of how far ahead in time the algorithm looks
                     # might not be good now
                     # To prioritise rewards in the distant future, the value is kept one
                     # deciding whether or not we want to value current rewards or future rewards
epsilon = 1.0        # exploration factor starting from one
                     # Exploration : Choose a uniformly random choice, random force to use; agent choosing an alg thin
king it
                     # will have the best long term effect
                     # avoid local minimum
                     # exploitation: when you keep doing what you were doing; exploration: when you try something new
epsilon_min = 0.01   # starting with high expl with and then immediately start lowering this
epsilon_decay = 0.995 # how quickly it will stop exploring
alpha = 0.01         # Learning rate: how big you take a leap in finding optimal policy
                     # it will determine to what extent new info will override old info
                     # alpha=0 means no learning; alpha = 1 means considering only recent info
alpha_decay = 0.01   # lowering alpha

batch_size = 64      # 64 samples
monitor = False      # stuff for OpenAI
quiet = False        # control print statements


# Environment Parameters

# for AI Gym

memory = deque(maxlen = 100000)    # custom list parameter, setting(controlling) max length
env = gym.make("CartPole-v0")
if max_env_steps is not None:
    env.max_episode_steps = max_env_steps
```

## Building the neural network

In [8]:
```
# building the neural network

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam


#Model definition

model = Sequential()
model.add(Dense(24, input_dim=4, activation = 'relu'))
                # 24 neurons, input dimensions = 4 as current environment has 4 paramters
                # activation is rectified linear unit

#adding hidden layers

model.add(Dense(48, activation = 'relu'))
model.add(Dense(2, activation = 'relu'))
                # we have force to the left and to the right
                # so two possible outputs; so 2 neurons

#how to compile this
model.compile(loss = 'mse', optimizer = Adam(lr = alpha, decay = alpha_decay))    # learning rate is alpha
```

## Defining necessary functions

In [9]:

```
# defining necessary functions

#setting up memory
def remember(state, action, reward, next_state, done):          # reward that we got, checking whether it is done ot n
ot
    memory.append((state, action, reward, next_state, done))

#choose action: pick what to do
def choose_action(state, epsilon):
    return env.action.sample() if (np.random.random() <= epsilon) else np.argmax(model.predict(state))
                                                #if no. chosen randomly from action space <= 1(at start)
                                                #if not, we shall get our model making up prediction based off current
state
                                                            #i.e., for exploration stage, prediction on force and dire
ction

def get_epsilon(t):
    return max(epsilon_min, min(epsilon, 1.0-math.log10((t+1)*epsilon_decay)))
                                                #towards the end we'd be decreasing substantially
                                                # in the beginning, right up at epsilon

# getting preprocess
def preprocess_state(state):
    return np.reshape(state, [1, 4])            # transposing state matrix to a column

#going through replay
def replay(batch_size, epsilon):
    x_batch, y_batch = [], []
    minibatch = random.sample(memory, min(len(memory), batch_size))

    for state, action, reward, next_state, done in minibatch:
        y_target = model.predict(state)
        y_target[0][action] = reward if done else reward + gamma + np.max(model.predict(next_state)[0])
        x_batch.append(state[0])
        y_batch.append(y_target[0])

    #fit our model
    #using the actions to train our model
    model.fit(np.array(x_batch), np.array(y_batch), batch_size=len(x_batch), verbose=0)
                                                                        #verbose: whther or not to make print statements o
utof this
```

```
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay
```

In [10]:

```
# define run function
# training our model which would choose the best action to do


def run():
    scores = deque(maxlen = 100)

    for e in range(n_episodes):
        state = preprocess_state(env.reset())    # start from the beginning each and everytime
        done = False
        i = 0                                    # time-set = 0

        while not done:                          # while done is false
            action = choose_action(state, get_epsilon(e))
            next_state, reward, done, _ = env.step(action)
            env.render()                         # rendering so that we can see what's goin' on
            next_state = preprocess_state(next_state)
            remember(state, action, reward, next_state, done)
            state = next_state
            i += 1

        scores.append(i)

        mean_score = np.mean(scores)

        if mean_score >= n_win_ticks and e >= 100:
            if not quiet: print('Ran {} episodes. Solved after {} trials'.format(e, e-100))
            return e-100
        if e % 20 == 0 and not quiet:
            print('[episode {}] - Mean survival time over last 100 episodes was {} ticks.'.format(e, mean_score))


        replay(batch_size,epsilon)

    if not quiet: print('did not solve after {} episodes'.format(e))
    return e
```

## Training the network

In [ ]:

```
# copying and pasting all the things from above

# as running the environment already initiated is not a good idea




import gym
import keras
import random
import math
import numpy as np
from collections import deque



# training parameters

n_episodes = 1000    # no. of episodes
n_win_ticks = 195    # every time step is a tick(in OpenAI); done state = win_tick
max_env_steps = None # for OPen AI

# RL parameters

gamma = 1.0          # Discount factor: measure of how far ahead in time the algorithm looks
                     # might not be good now
                     # To prioritise rewards in the distant future, the value is kept one
                     # deciding whether or not we want to value current rewards or future rewards
epsilon = 1.0        # exploration factor starting from one
                     # Exploration : Choose a uniformly random choice, random force to use; agent choosing an alg thin
king it
                     # will have the best long term effect
                     # avoid local minimum
                     # exploitation: when you keep doing what you were doing; exploration: when you try something new
epsilon_min = 0.01   # starting with high expl with and then immediately start lowering this
epsilon_decay = 0.995 # how quickly it will stop exploring
alpha = 0.01         # Learning rate: how big you take a leap in finding optimal policy
                     # it will determine to what extent new info will override old info
                     # alpha=0 means no learning; alpha = 1 means considering only recent info
```

```
alpha_decay = 0.01   # lowering alpha

batch_size = 64      # 64 samples
monitor = False      # stuff for OpenAI
quiet = False        # control print statements


# Environment Parameters

# for AI Gym

memory = deque(maxlen = 100000)    # custom list parameter, setting(controlling) max length
env = gym.make("CartPole-v0")
if max_env_steps is not None:
    env.max_episode_steps = max_env_steps


# building the neural network

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam


#Model definition

model = Sequential()
model.add(Dense(24, input_dim=4, activation = 'relu'))
                # 24 neurons, input dimensions = 4 as current environment has 4 paramters
                # activation is rectified linear unit

#adding hidden layers

model.add(Dense(48, activation = 'relu'))
model.add(Dense(2, activation = 'relu'))
                # we have force to the left and to the right
                # so two possible outputs; so 2 neurons

#how to compile this
model.compile(loss = 'mse', optimizer = Adam(lr = alpha, decay = alpha_decay))    # learning rate is alpha
```

```
# defining necessary functions

#setting up memory
def remember(state, action, reward, next_state, done):          # reward that we got, checking whether it is done ot n
ot
    memory.append((state, action, reward, next_state, done))

#choose action: pick what to do
def choose_action(state, epsilon):
    return env.action_space.sample() if (np.random.random() <= epsilon) else np.argmax(model.predict(state))
                                                #if no. chosen randomly from action space <= 1(at start)
                                                #if not, we shall get our model making up prediction based off current
state
                                                            #i.e., for exploration stage, prediction on force and dire
ction

def get_epsilon(t):
    return max(epsilon_min, min(epsilon, 1.0-math.log10((t+1)*epsilon_decay)))
                                                #towards the end we'd be decreasing substantially
                                                # in the beginning, right up at epsilon


# getting preprocess
def preprocess_state(state):
    return np.reshape(state, [1, 4])            # transposing state matrix to a column

#going through replay
def replay(batch_size, epsilon):
    x_batch, y_batch = [], []
    minibatch = random.sample(memory, min(len(memory), batch_size))

    for state, action, reward, next_state, done in minibatch:
        y_target = model.predict(state)
        y_target[0][action] = reward if done else reward + gamma + np.max(model.predict(next_state)[0])
        x_batch.append(state[0])
        y_batch.append(y_target[0])

    #fit our model
    #using the actions to train our model
    model.fit(np.array(x_batch), np.array(y_batch), batch_size=len(x_batch), verbose=0)
                                                                        #verbose: whther or not to make print statements o
utof this
```

```
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay


# define run function
# training our model which would choose the best action to do


def run():
    scores = deque(maxlen = 100)

    for e in range(n_episodes):
        state = preprocess_state(env.reset())    # start from the beginning each and everytime
        done = False
        i = 0                                    # time-set = 0

        while not done:                          # while done is false
            action = choose_action(state, get_epsilon(e))
            next_state, reward, done, _ = env.step(action)
            env.render()                         # rendering so that we can see what's goin' on
            next_state = preprocess_state(next_state)
            remember(state, action, reward, next_state, done)
            state = next_state
            i += 1

        scores.append(i)

        mean_score = np.mean(scores)

        if mean_score >= n_win_ticks and e >= 100:
            if not quiet: print('Ran {} episodes. Solved after {} trials'.format(e, e-100))
            return e-100
        if e % 20 == 0 and not quiet:
            print('[episode {}] - Mean survival time over last 100 episodes was {} ticks.'.format(e, mean_score))


        replay(batch_size,epsilon)

    if not quiet: print('did not solve after {} episodes'.format(e))
    return e
```

```
run()
```

```
[episode 0] - Mean survival time over last 100 episodes was 11.0 ticks.
[episode 20] - Mean survival time over last 100 episodes was 105.61904761904762 ticks.
[episode 40] - Mean survival time over last 100 episodes was 67.1219512195122 ticks.
[episode 60] - Mean survival time over last 100 episodes was 48.22950819672131 ticks.
[episode 80] - Mean survival time over last 100 episodes was 38.617283950617285 ticks.
[episode 100] - Mean survival time over last 100 episodes was 38.44 ticks.
[episode 120] - Mean survival time over last 100 episodes was 18.36 ticks.
[episode 140] - Mean survival time over last 100 episodes was 15.09 ticks.
[episode 160] - Mean survival time over last 100 episodes was 16.26 ticks.
[episode 180] - Mean survival time over last 100 episodes was 18.35 ticks.
[episode 200] - Mean survival time over last 100 episodes was 13.74 ticks.
[episode 220] - Mean survival time over last 100 episodes was 14.76 ticks.
[episode 240] - Mean survival time over last 100 episodes was 18.48 ticks.
[episode 260] - Mean survival time over last 100 episodes was 18.55 ticks.
[episode 280] - Mean survival time over last 100 episodes was 17.91 ticks.
[episode 300] - Mean survival time over last 100 episodes was 22.76 ticks.
[episode 320] - Mean survival time over last 100 episodes was 29.01 ticks.
[episode 340] - Mean survival time over last 100 episodes was 31.33 ticks.
[episode 360] - Mean survival time over last 100 episodes was 32.01 ticks.
[episode 380] - Mean survival time over last 100 episodes was 33.71 ticks.
[episode 400] - Mean survival time over last 100 episodes was 31.25 ticks.
[episode 420] - Mean survival time over last 100 episodes was 26.62 ticks.
[episode 440] - Mean survival time over last 100 episodes was 23.28 ticks.
[episode 460] - Mean survival time over last 100 episodes was 23.83 ticks.
[episode 480] - Mean survival time over last 100 episodes was 23.14 ticks.
[episode 500] - Mean survival time over last 100 episodes was 22.76 ticks.
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
_ctypes/callbacks.c in 'calling callback function'()

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\win32\__init__.py in f(hwnd, msg, wParam, lParam)
    637             if event_handler:
    638                 if self._allow_dispatch_event or not self._enable_event_queue:
--> 639                     result = event_handler(msg, wParam, lParam)
    640                 else:
    641                     result = 0

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\win32\__init__.py in _event_key(self, msg, wParam, lParam)
    696
    697         if symbol is None:
--> 698             symbol = key.user_key(wParam)
    699         elif symbol == key.LCTRL and lParam & (1 << 24):
    700             symbol = key.RCTRL

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\key.py in user_key(scancode)
    179     (for example, mapping keys to actions in a game options screen).
    180     """
--> 181     assert scancode > 0
    182     return scancode << 32
    183

AssertionError:
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
_ctypes/callbacks.c in 'calling callback function'()

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\win32\__init__.py in f(hwnd, msg, wParam, lParam)
    637             if event_handler:
    638                 if self._allow_dispatch_event or not self._enable_event_queue:
--> 639                     result = event_handler(msg, wParam, lParam)
    640                 else:
    641                     result = 0

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\win32\__init__.py in _event_key(self, msg, wParam, lParam)
    696
    697         if symbol is None:
--> 698             symbol = key.user_key(wParam)
    699         elif symbol == key.LCTRL and lParam & (1 << 24):
    700             symbol = key.RCTRL

C:\ProgramData\Anaconda3\lib\site-packages\pyglet\window\key.py in user_key(scancode)
    179     (for example, mapping keys to actions in a game options screen).
    180     """
--> 181     assert scancode > 0
    182     return scancode << 32
    183

AssertionError:

[episode 520] - Mean survival time over last 100 episodes was 22.78 ticks.
[episode 540] - Mean survival time over last 100 episodes was 22.32 ticks.
```