```
import numpy as np
import gym
import matplotlib.pyplot as plt


env = gym.make("CartPole-v1")
state = env.reset()
state
# position, velocity, angle, angular velocity
```

```
array([-0.0405212 ,  0.03900573, -0.04038244,  0.04199292])
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from collections import deque
import random
import tensorflow as tf


model = Sequential()
model.add(Dense(units = 50, input_dim=4, activation='relu'))
model.add(Dense(units = 50, activation = "relu"))
model.add(Dense(units = 2, activation = "linear"))


opt = tf.keras.optimizers.Adam(learning_rate=0.001)
#opt = tf.keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='MSE',optimizer=opt)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                250

 dense_1 (Dense)             (None, 50)                2550
```

```
 dense_2 (Dense)             (None, 2)                 102

=================================================================
Total params: 2,902
Trainable params: 2,902
Non-trainable params: 0
_________________________________________________________________
```

Parametry:

```
train_episodes = 100
epsilon = 0.16
gamma = 0.99
max_steps = 50
```

Definiujemy **pamięć** jako kolejkę:

```
memory = deque(maxlen=50)
```

Ustalamy rozmiar **batch**:

```
batch_size = 5


def train():
  state_batch, Qs_target_batch = [], []

  minibatch = random.sample(memory, batch_size)

  for state, action, reward, next_state, done in minibatch:

    if done:
      y = reward
    else:
      y = reward + gamma*np.max(model.predict(next_state)[0])
```

```
      Q_target = model.predict(state)
      Q_target[0][action] = y

      state_batch.append(state)
      Qs_target_batch.append(Q_target)

    state_batch = np.array(state_batch).reshape(batch_size,4)
    Qs_target_batch = np.array(Qs_target_batch).reshape(batch_size,2)

    h = model.fit(state_batch,Qs_target_batch,epochs=1,verbose=0)

    loss = h.history['loss'][0]

    return loss

  Loss = []
  Rewards = []

  for e in range(1, train_episodes+1):
    total_reward = 0
    t = 0

    state = env.reset()
    state = np.reshape(state, [1, 4])

    done = False
    while t < max_steps and done == False:

      Qs = model.predict(state)[0]

      if np.random.rand()<epsilon:
        action = env.action_space.sample()
      else:
        action = np.argmax(Qs)

      next_state, reward, done, _ = env.step(action)
      next_state = np.reshape(next_state, [1, 4])

      total_reward += reward

      memory.append((state,action,reward,next_state,done))
```

```
      memory.append((state,action,reward,next_state,done))

      if batch_size < len(memory):
        loss = train()
        Loss.append(loss)

      state = next_state
      t+=1

    print(e," R=",total_reward)
    Rewards.append(total_reward)
```

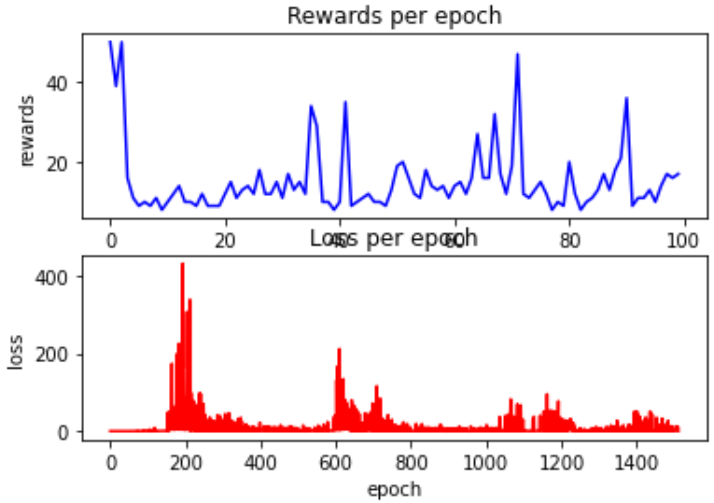
```
 43  R= 9.0
 44  R= 10.0
 45  R= 11.0
 46  R= 12.0
 47  R= 10.0
 48  R= 10.0
 49  R= 9.0
 50  R= 13.0
 51  R= 19.0
 52  R= 20.0
 53  R= 16.0
 54  R= 12.0
 55  R= 11.0
 56  R= 18.0
 57  R= 14.0
 58  R= 13.0
 59  R= 14.0
 60  R= 11.0
 61  R= 14.0
 62  R= 15.0
 63  R= 12.0
 64  R= 16.0
 65  R= 27.0
 66  R= 16.0
 67  R= 16.0
 68  R= 32.0
 69  R= 17.0
 70  R= 12.0
 71  R= 19.0
 72  R= 47.0
 73  R= 12.0
 74  R= 11.0
 75  R= 13.0
```

```
76  R= 15.0
77  R= 12.0
78  R= 8.0
79  R= 10.0
80  R= 9.0
81  R= 20.0
82  R= 12.0
83  R= 8.0
84  R= 10.0
85  R= 11.0
86  R= 13.0
87  R= 17.0
88  R= 13.0
89  R= 18.0
90  R= 21.0
91  R= 36.0
92  R= 9.0
93  R= 11.0
94  R= 11.0
95  R= 13.0
96  R= 10.0
97  R= 14.0
98  R= 17.0
99  R= 16.0
100  R= 17.0
```

```
plt.subplot(211)
plt.ylabel('rewards')
plt.title('Rewards per epoch')
plt.plot(range(len(Rewards)),Rewards,"b")

plt.subplot(212)
plt.xlabel('epoch')
plt.ylabel('loss')
plt.title('Loss per epoch')
plt.plot(range(len(Loss)),Loss,"r")

plt.show()
```



✓ 0 s ukończono o 23:12