CartPole-v1 with TensorFlow2

Problem gym.openai.com/envs/CartPole-v1/

Setup

```
import numpy as np
import os
import pandas as pd
import random
```

```
try:
    import gym
except:
    !pip install gym
import gym
print(gym.__version__)
```

```
0.10.11
```

```
try:
    import tensorflow as tf
except:
    !pip install tensorflow==2.0.0-alpha0
    import tensorflow as tf
if tf.__version__[0] == "1":
    !pip install tensorflow==2.0.0-alpha0
    import tensorflow as tf
print(tf.__version__)
```

```
2.0.0-alpha0
```

Visualize gym environment - random action

```
import gym
env = gym.make("CartPole-v1")
no_actions = env.action_space.n
no_observations = env.observation_space.shape[0]
print(no_actions)
print(no_observations)
```

```
2
4
```

Action and observation interpretations:
github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py

```
observation = env.reset()
```

```
for t in range(100):
#     env.render()
    action = env.action_space.sample()
    observation, reward, done, info = env.step(action)
    if done:
        print("Episode finished after {} timesteps".format(t+1))
        break
env.close()
```

```
Episode finished after 13 timesteps
```

Train with TF2, Keras

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(
    24,
    activation="relu",
    input_shape=(no_observations,)))
model.add(tf.keras.layers.Dense(24, activation="relu"))
model.add(tf.keras.layers.Dense(no_actions, activation="linear"))
model.compile(loss="mse", optimizer=tf.keras.optimizers.Adam(lr=0.0
```

```
def get_best_action(state):
    state_df = pd.DataFrame(columns=["cart_pos", "cart_vel", "pole_
    state_df.loc[len(state_df),:] = state
    q = model.predict(state_df.values, batch_size=1)
    action = pd.Series(q[0]).idxmax()
    return action, q
```

```
GAMMA = 0.95
def fit_model(state, action, reward, next_state, done):
    state_action, state_q = get_best_action(state)
    next_state_action, next_state_q = get_best_action(next_state)
    q_update = reward
    if not done:
        q_update = (reward + GAMMA * next_state_q[0][next_state_act
    state_q[0][action] = q_update
    state_df = pd.DataFrame(columns=["cart_pos", "cart_vel", "pole_
    state_df.loc[len(state_df),:] = state
    model.fit(state_df.values, state_q, epochs=1, batch_size=1)
```

```
%%capture
cap_T = 1500
for ep in range(cap_T):
    state = env.reset()
    for t in range(500):
#         env.render()
        if random.random() < (1 - ep / cap_T):
            action = env.action_space.sample()
        else:
            action, q = get_best_action(state)
        next_state, reward, done, info = env.step(action)
        fit_model(state, action, reward, next_state, done)
        state = next_state
        if done:
            print("Episode finished after {} timesteps".format(t+1)
            break
```

Test with TF2, Keras

```python
# env.render()
for ep in range(10):
    state = env.reset()
    for t in range(1000):
        action, q = get_best_action(state)
        next_state, reward, done, info = env.step(action)
        if done:
            print("Episode {} finished after {} timesteps".format(e
            break
        state = next_state
env.close()
```

```
Episode 0 finished after 99 timesteps
Episode 1 finished after 101 timesteps
Episode 2 finished after 103 timesteps
Episode 3 finished after 104 timesteps
Episode 4 finished after 99 timesteps
Episode 5 finished after 103 timesteps
Episode 6 finished after 104 timesteps
Episode 7 finished after 106 timesteps
Episode 8 finished after 102 timesteps
Episode 9 finished after 105 timesteps
```