Μηχανική Μάθηση

Final Project 2023

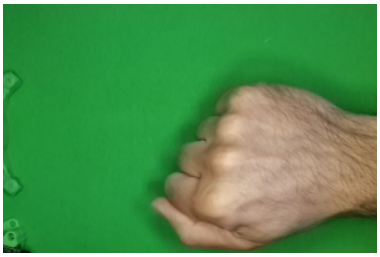
Rock-Scissors-Paper Agent

Περιγραφή

Στόχος του project είναι η κατασκευή ενός ευφυούς πράκτορα που θα μάθει να παίζει το παιχνίδι Rock-Scissor-paper. Ποιο συγκεκριμένα, ο πράκτορας θα βλέπει μία εικόνα που αντιστοιχεί σε 0: Πέτρα (Rock), 1: Ψαλίδι (Scissor) ή 2: Χαρτί (Paper) και θα επιλέγει το αντίστοιχο σύμβολο που το κερδίζει.

Για την εκπαίδευση του πράκτορα σας, μπορείτε να χρησιμοποιήσετε το παρακάτω σύνολο δεδομένων: https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors

το οποίο περιέχει 700+ εικόνες για κάθε κίνηση


Οδηγίες Παιχνιδιού

Ο δικός σας πράκτορας θα ποντάρει 1€ εναντίων ενός "Random Agent" για συνολικά N γύρους. Αν ο πράκτορας σας κερδίσει, του επιστρέφονται 2€, στην ισοπαλία του επιστρέφεται 1€, διαφορετικά έχει χάσει 1€. Ο Random Agent θα παίζει πάντα 1^{ος}, διαλέγοντας μία τυχαία εικόνα από τις 2100 εικόνες συνολικά (μπορεί να είναι είτε χαρτί, είτε πέτρα, είτε ψαλίδι). **Αντί για τις 2100 εικόνες, μπορείτε να χρησιμοποιήσετε ένα ποσοστό του dataset ως test set, όπως περιγράφεται παρακάτω.**

Επειδή ο Random Agent είναι πονηρός, μπορεί να εφαρμόσει vertical flip με πιθανότητα $p_1 = 0.5$ και ύστερα με πιθανότητα $p_2 = 0.5$ μπορεί εφαρμόζει horizontal flip. Τέλος, ο Random Agent εφαρμόζει τυχαίο θόρυβο (white noise) σε κάθε pixel της εικόνας με μέση τιμή 0 και τυπική απόκλιση 5% της μέγιστης τιμής του pixel. Η λειτουργία του Random Agent μπορεί να περιγραφεί ως:

1. Αντιστοιχίστε τις κινήσεις (πέτρα, ψαλίδι, χαρτί) σε (0, 1, 2 αντίστοιχα).
2. **Χωρίστε το dataset σε train-test: Μπορείτε για κάθε κλάση να επιλέξετε ένα ποσοστό ως test set (πχ 20% της πέτρας, 20% του ψαλιδιού και 20% του χεριού) και να χρησιμοποιήσετε αυτά για τη δοκιμή του μοντέλου/πράκτορα σας.**

3. Select Image: Επέλεγε τυχαία μία εικόνα από τις 2100 (που αντιστοιχεί σε κίνηση 0, είτε 1, είτε 2).
4. Preprocess Image: Επεξεργάσου την εικόνα:
   a. Με πιθανότητα p_1 εφάρμοσε Vertical Flip
   b. Με πιθανότητα p_2 εφάρμοσε Horizontal Flip
   c. Πρόσθεσε θόρυβο με $\mu = 0,\ \sigma = 255 * 0.05$). Αν κανονικοποιήσετε την εικόνα, προσαρμόστε την τυπική απόκλιση ανάλογα.
   d. Μπορείτε να εφαρμόσετε όποια άλλη μέθοδο Image Processing θέλετε, ώστε να δυσκολέψετε το παίχνιδι (αν το επιθυμείτε).
5. Apply: Στείλτε την εικόνα στον πράκτορα σας.
6. Ο πράκτορας διαβάζει την εικόνα και επιλέγει την βέλτιστη ενέργεια.
7. Στόχος είναι η μεγιστοποίηση του κέρδους. Άρα, θα χρειαστεί να κάνετε plot το κέρδος του πράκτορα (Μπορείτε να αποθηκεύσετε το συνολικό κέρδος σε κάθε γύρο και να το πλοτάρετε στο τέλους του παιχνιδιού).
8. Ως τελικό στόχο, δοκιμάστε την ακρίβεια του πράκτορα (ή μοντέλου) σας σε εικόνες εκτός του dataset, πχ από το internet ή δικές σας, τις οποίες θα πρέπει να κάνετε rescaling στο ίδιο μέγεθος. Για παράδειγμα, Μπορείτε να δοκιμάσετε την ενέργεια που θα βγάλει για την παρακάτω εικόνα.

Ως παραδοτέο, ανεβάστε το Google Colab, μέσα στο οποίο θα υπάρχει (πάνω πάνω) κάποιος σύνδεσμος για τη πλατφόρμα του github, όπου θα έχετε ανεβάσει και εκεί πέρα στο σημειωματάριο σας.

Χρήσιμες Συμβουλές

1. **Dimensionality Reduction:** Οι εικόνες έχουν μεγάλο μέγεθος. Μπορείτε να μικραίνετε το μέγεθος αυτό (πχ 30x30). Προσέξτε το (ratio) width/height, ώστε να μην αλλειωθεί το μέγεθος της εικόνας κατά τη σμίκρυνση. Μπορείτε να χρησιμοποιήσετε όποια βιβλιοθήκη θέλετε (numpy, opencv, κλπ.)
2. **Dimensionality Reduction**: Μπορείτε να μετατρέψετε τις εικόνες σε ασπρόμαυρες ώστε να μειώσετε το σύνολο των pixels στο 1/3.
3. **Normalization**: Μπορείτε να κανονικοποιήσετε τις εικόνες στο 0-1.
4. **Environment**: Θα χρειαστεί να κατασκευάσετε ένα απλοικό environment με όποιο τρόπο σας βολε
5. ύει (For/While loop, Function, Class, etc.) για τη δημιουργία του παιχνιδιού.

6. **Μέθοδοι Επίλυσης (<u>Διαλέξτε έναν</u>)**:
   a. **Supervised Learning**: Θα χρειαστεί να εκπαιδεύσετε κάποιο μοντέλο μηχανικής μάθησης (πχ Logistic Regression, Random Forest, Convolutional Neural Network, Deep Neural Network, SVM, KNN, κλπ.) για να αναγνωρίζετε τις εικόνες. Έπειτα, αφού αναγνωρίσετε την εικόνα, μπορείτε να επιλέξετε την καλύτερη δυνατή ενέργεια με κάποιον κανόνα (πχ Αν εικόνα == «Πέτρα», τότε ενέργεια = «Χαρτί».
   b. **Deep Reinforcement Learning**: Μπορείτε (καθαρά για λόγους εξάσκησης) να αυτοματοποιήσετε πλήρως τη διαδικασία χρησιμοποιόντας πράκτορα DRL (πχ PPO agent), ο οποίος θα δέχεται ως είσοδο την εικόνα και θα βγάζει την βέλτιστη ενέργεια (0, 1,ή 2). Μπορείτε να πειράξετε την παράμετρο "*conv_filters*" του config.model_config , ώστε να ορίσετε αριθμό Convolution φίλτρων (αν επιθυμείτε). Για παράδειγμα *conv_filters=[(64, 5, 1), (64, 5,1)]* χρησιμοποιεί 2 επίπεδα 64 φίλτρων μεγέθους $5x5$ και $strides = 1$. Επιπλέον θα χρειαστεί να φτιάξετε κάποιο Custom Gym Environment, το οποίο θα χρησιμοποιήσει το rllib, υλοποιόντας τις συναρτήσεις reset, step.

```python
import gymnasium as gym
from gym import spaces
import numpy as np


class RockPaperScissorsEnv(gym.Env):
    def __init__(self, env_config: dict):
        super().__init__()

        self.rock_images = env_config['rock_images']
        self.scissor_images = env_config['scissor_images']
        self.paper_images = env_config['paper_images']

        self.action_space = spaces.Discrete(3)  # States: 0: Rock, 1: Paper, 2: Scissors
        self.observation_space = spaces.Box(low=0, high=1.0, shape=(28, 28, 1), dtype=np.float32)  # Normalized grayscaled image of rock/scissor/paper
        self.total_wins = 0
        self.wins = [0]

        self.image = None
        self.state = None

    def select_image(self):
        pass
        # Implement a function that selects random a state (0/1/2 and a random image of that state)

    def preprocess_image(self, image: np.ndarray) -> np.ndarray:
        # Implement a function that preprocesses the image (e.g. add input noise, rotates the image, etc.)

    def reset(self, seed=None, options=None):
        self.image, self.state = get_random_image
        return self.image

    def step(action: int) -> tuple:
        if action == 0 and self.state == 0:
            reward = 0
        # Compute the reward based on the agent's selection
        # elif action == ... and state = ...

        next_state = self.image # It doesn't matter what is the next state, the round ends anyway.
        done = True
        truncated = False
        info = {}
        return self.image, reward, done, truncated, info
```

Στη συνέχεια, μπορείτε να το κάνετε register, όπως παρακάτω:

```python
from gym.envs.registration import register

register(
    id='gym_examples/GridWorld-v0',
    entry_point='gym_examples.envs:GridWorldEnv',
    max_episode_steps=300,
)
```

```python
import gym_examples
env = gym.make('gym_examples/GridWorld-v0')
```

Απαιτήσεις

1. Να ανεβάσετε τον κώδικα σας (.ipynb) εκτελεσμένο στην πλατφόρμα "**Github**" https://github.com/ με ένα επαρκές readme (πχ https://github.com/kochlisGit/Shadow-Hand-Controller). Βάλτε το link του github στο ipynb ώστε να είναι ορατό.
2. Πειραματιστείτε με διάφορα μοντέλα, μεθόδους επεξεργασίας εικόνων, τεχνικές βελτίωσης της ακρίβειας των μοντέλων/πρακτόρων κλπ. Μπορείτε να υποβάλλετε την άσκηση με τη χρήση μόνο ενός μοντέλου/πράκτορα, αλλά να αιτιολογήσετε το λόγο που το επιλέξατε (πχ το επελεξα γιατί φαίνεται πως έχει απόδοση 99%).
3. Κάντε ανάλυση σε ποιες εικόνες δουλεύει καλά και σε ποιες όχι η τελική σας λύση, καθώς και με ποιες τεχνικές το αντιμετωπίσατε.
4. Αιτιολογήστε τα μοντέλα και τις τεχνικές που επιλέξατε, καθώς και την απόδοση σας.