To customize your agent to kick the ball in the smart soccer ball project, you will need to modify the environment and training configuration to recognize kicking actions and train the agent for this specific task. Here's a step-by-step guide based on your provided code:

**1. Modify the Environment to Include Kicking Mechanics**

In your customenv.py, the ball and goal are already defined, but there is no explicit handling for kicking. You'll need to:

* Define a new action for kicking.
* Update the step function to handle this action and update the ball's position accordingly.

class CustomMazeSoccerBallEnv(MiniGridEnv):

def \_\_init\_\_(self, size=25):

super().\_\_init\_\_(grid\_size=size, max\_steps=4 \* size \*\* 2, render\_mode='human')

self.action\_space = Discrete(5) # Add an extra action for 'kick'

def step(self, action):

if action == 4: # Kick action

ball\_pos = self.ball.cur\_pos

agent\_pos = self.agent\_pos

# Kick the ball towards the goal if adjacent

if self.\_distance(agent\_pos, ball\_pos) == 1:

self.\_move\_ball\_towards\_goal()

else:

obs, reward, terminated, truncated, info = super().step(action)

if self.ball.cur\_pos == self.goal.cur\_pos:

reward = 1

terminated = True

return obs, reward, terminated, truncated, info

def \_move\_ball\_towards\_goal(self):

# Move ball one step closer to the goal

goal\_pos = self.goal.cur\_pos

dx = np.sign(goal\_pos[0] - self.ball.cur\_pos[0])

dy = np.sign(goal\_pos[1] - self.ball.cur\_pos[1])

new\_pos = (self.ball.cur\_pos[0] + dx, self.ball.cur\_pos[1] + dy)

if self.grid.get(\*new\_pos) is None: # Ensure no obstacles

self.grid.set(\*self.ball.cur\_pos, None) # Clear previous position

self.grid.set(\*new\_pos, self.ball)

self.ball.cur\_pos = new\_pos

**2. Update the Action Space in the Observation Wrapper**

Ensure your CustomObservationWrapper or any other observation wrapper you are using accounts for the new action.

**3. Train the Agent**

Update the train.py file to include the kicking action during training. The agent will learn to kick the ball strategically towards the goal.

model = PPO(

"MlpPolicy",

env,

verbose=1,

learning\_rate=3e-4,

n\_steps=1024,

batch\_size=64,

tensorboard\_log="./logs/tensorboard",

)

**No changes** are required here as the PPO algorithm will adapt to the updated action space.

**4. Test the Agent**

Update your test\_agent function in test.py to validate the agent's ability to kick the ball successfully.

def test\_agent(model, env):

obs = env.reset()

done = False

while not done:

env.render()

action, \_ = model.predict(obs)

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

print(f"Action: {action}, Reward: {reward}")

**5. Visualization and Feedback**

Incorporate real-time visualization using env.render() during training and testing to debug the agent's kicking behavior.

This setup will allow your smart soccer ball project to incorporate kicking mechanics effectively.