Humanoid

1. **The Original Problem**: Understanding the issues with the original - The goal of teaching a humanoid robot to run in a reinforcement learning (RL) environment.
    - The bug in the original reward function and other limitations preventing running behavior.
2. **Modifications to the Code**: A detailed breakdown of each change made to humanoid_v5_custom-Copy3.py, including the rationale and expected impact on the humanoid's behavior.
3. **Pushing to GitHub**: The process of committing and pushing the modified file to the GYM_humanoid repository, including resolving the Git error (fatal: pathspec 'humanoid_v5_custom-Copy3.py' did not match any files).
4. **Visualization and Summary**: A clear visualization of the entire process, connecting the code changes to the Git workflow and the RL goal.
5. **Next Steps**: Guidance on testing the modified environment, integrating it with RL training, and collaborating with your team.

I'll explain each part carefully, ensuring all details are clear, even for someone new to RL, MuJoCo, or Git. I'll reference the provided code (humanoid_v5_custom-Copy3.py) to ground the explanation in the actual changes made. Let's dive in!

1. The Original Problem

Context

You were working on a reinforcement learning project using the Gymnasium Humanoid-v5 environment, a MuJoCo-based simulation of a 3D bipedal robot designed to mimic a human. The robot has a torso, arms, legs, and tendons, with 17 hinge joints controlled by torque actions. Your goal was to train the humanoid to **run forward as fast as possible** without falling, using an RL algorithm (e.g., PPO, SAC).

Issues with the Original Environment

The original humanoid_v5_custom-Copy3.py (based on Humanoid-v5) had several issues that prevented the humanoid from learning to run:

1. **Bug in the Reward Function**:
    - In the _get_rewmethod, the reward was calculated as:
    - rewards = healthy_reward - forward_reward
        - The forward_reward was defined as forward_reward_weight * x_velocity (with forward_reward_weight = 1.25), intended to reward forward movement.
        - **Problem**: Subtracting forward_reward penalized forward movement, discouraging the humanoid from moving faster. This was a critical bug, as it contradicted the goal of running.
2. **Insufficient Incentive for High Speed**:

    - The forward_reward was linear (1.25 * x_velocity), which didn't strongly reward high velocities needed for running.
    - The healthy_reward (7.0 per step when the humanoid was upright) dominated the reward, encouraging the humanoid to stand still to maximize this reward rather than move forward.
3. **Overly Restrictive Control Cost**:
    - The ctrl_cost was calculated as ctrl_cost_weight * sum(square(data.ctrl)) with ctrl_cost_weight = 0.1.
    - This penalized large actions (joint torques), discouraging the dynamic, high-torque movements required for running.
4. **Ineffective Contact Cost**:
    - The contact_cost was contact_cost_weight * clip(sum(square(cfrc_ext)), min_cost, max_cost) with contact_cost_weight = 5e-7.
    - Due to a known bug in Humanoid-v4 (and possibly v5), cfrc_ext (external contact forces) was often zero, making this term ineffective.
    - Even if non-zero, the small weight (5e-7) made the penalty negligible.
5. **Lack of Running-Specific Rewards**:
    - The reward function didn't incentivize behaviors specific to running, such as:
        - Upright posture (e.g., minimal torso tilt).
        - High foot clearance during the swing phase.
        - Alternating leg movements for a natural gait.
    - This could lead to suboptimal gaits (e.g., hopping, crawling) instead of running.
6. **Termination and Stability**:
    - The environment terminated when the humanoid was "unhealthy" (torso z-coordinate outside [1.0, 2.0]), which was appropriate but didn't penalize falling strongly enough to encourage robust running.

Impact

- The humanoid likely stood still or moved slowly to maximize healthy_reward while avoiding penalties from forward_reward and ctrl_cost.
- The lack of strong incentives for high velocity and running-specific behaviors made it impossible for the RL agent to learn a running gait.

Goal

Fix the reward function and environment parameters to:

- Reward high forward velocity.
- Encourage stability and upright posture.
- Allow dynamic actions for running.
- Promote natural running behaviors.
- Share the modified environment with your team via the GYM_humanoid GitHub repository for collaborative development.

2. Modifications to the Code

We modified humanoid_v5_custom-Copy3.py to address the above issues. Below, I'll explain each change in the provided code, referencing specific lines or sections, and detail why it was made and its expected effect. The changes are grouped by the methods or sections they affect: __init__, _get_rew, step, and minor corrections.

2.1. __init__ Method (Initialization)

Code Changes:

```
def __init__(
    self,
    xml_file: str = "humanoid.xml",
    frame_skip: int = 5,
    default_camera_config: Dict[str, Union[float, int]] = DEFAULT_CAMERA_CONFIG,
    forward_reward_weight: float = 5.0,  # changed from 1.25, increase to prioritize running
    ctrl_cost_weight: float = 0.05,     # reducing, allows dynamic actions
    contact_cost_weight: float = 5e-7,
    contact_cost_range: Tuple[float, float] = (-np.inf, 10.0),
    healthy_reward: float = 2.0,        # reduced to balance with forward_reward
    terminate_when_unhealthy: bool = True,
    healthy_z_range: Tuple[float, float] = (1.0, 2.0),
    reset_noise_scale: float = 1e-2,
    exclude_current_positions_from_observation: bool = True,
    include_cinert_in_observation: bool = True,
    include_cvel_in_observation: bool = True,
    include_qfrc_actuator_in_observation: bool = True,
    include_cfrc_ext_in_observation: bool = True,
    **kwargs,
)
```

Changes and Rationale:

1. **Increased forward_reward_weight**(from 1.25 to 5.0):
   - **Why**: To prioritize forward movement, making the forward_reward a dominant component of the total reward. A higher weight amplifies the reward for x_velocity, encouraging the humanoid to move faster, which is essential for running.
   - **Effect**: The RL agent will focus on maximizing x_velocity, leading to faster movement, potentially achieving running speeds.
2. **Reduced ctrl_cost_weight**(from 0.1 to 0.05):
   - **Why**: The original weight penalized large joint torques too heavily, restricting the dynamic actions needed for running (e.g., strong leg swings, rapid arm movements). Halving the weight allows more aggressive actions without excessive penalties.
   - **Effect**: The humanoid can apply larger torques, enabling the dynamic, high-energy movements required for running.
3. **Reduced healthy_reward**(from 7.0 to 2.0):
   - **Why**: The original healthy_reward was too large, dominating the reward and incentivizing the humanoid to stand still to remain "healthy" (torso z-coordinate in [1.0, 2.0]). Reducing it balances it with forward_reward, ensuring movement is prioritized.
   - **Effect**: The humanoid is still rewarded for staying upright but not at the expense of moving forward, encouraging a balance between stability and speed.
4. **Kept contact_cost_weight**(5e-7, unchanged):
   - **Why**: The contact_cost penalizes large external contact forces (cfrc_ext), but a bug in Humanoid-v4 (and possibly v5) often makes cfrc_ext zero, rendering this term ineffective. We kept it unchanged pending verification of whether cfrc_ext is non-zero in your setup.
   - **Effect**: If cfrc_ext is zero, this term has no impact. If non-zero, the small weight ensures minimal influence, avoiding unnecessary penalties during running.
5. **Other Parameters**(unchanged):
   - Parameters like frame_skip, healthy_z_range, reset_noise_scale, and observation settings were left as is, as they were appropriate for the environment.
   - **Why**: These settings define the simulation's timing, termination conditions, and observation space, which were already suitable for the RL task.

Impact:

- The initialization sets up a reward structure that prioritizes forward velocity (forward_reward_weight = 5.0), allows dynamic actions (ctrl_cost_weight = 0.05), and balances stability (healthy_reward = 2.0). This creates a foundation for the humanoid to learn running behavior.

2.2. _get_rew Method (Reward Function)

Code Changes:

```
def _get_rew(self, x_velocity: float, action):

  torso_angle = np.abs(self.data.qpos[3:7].sum())

  # Approximate foot z-coordinates

  right_foot_z = self.data.xpios[5,2]
```

```
    left_foot_z = self.data.xpios[8,2]


    forward_reward = self._forward_reward_weight * (x_velocity + 0.5 * x_velocity**2)
    healthy_reward = self.healthy_reward
    foot_clearance_reward = 2.0 * max(right_foot_z, left_foot_z) if max(right_foot_z, left_foot_z) > 0.2 else 0.0
    rewards = healthy_reward + forward_reward + foot_clearance_reward


    ctrl_cost = self.control_cost(action)
    contact_cost = self.contact_cost
    posture_penalty = -10.0 * torso_angle if torso_angle > 0.3 else 0.0
    costs = ctrl_cost + contact_cost + posture_penalty


    reward = rewards - costs


    reward_info = {
        "reward_survive": healthy_reward,
        "reward_forward": forward_reward,
        "reward_ctrl": -ctrl_cost,
        "reward_contact": -contact_cost,
        "reward_posture": -posture_penalty,
    }


    return reward, reward_info
```

Changes and Rationale:

1. **Fixed Forward Reward Bug**:
   - **Original**: rewards = healthy_reward - forward_reward.

- **Modified**: rewards = healthy_reward + forward_reward.
- **Why**: Subtracting forward_reward penalized forward movement, contradicting the goal of running. Adding it rewards forward velocity, aligning with the objective.
- **Effect**: The humanoid is now rewarded for moving forward, with the reward scaling with x_velocity.

2. **Added Quadratic Velocity Term**:
   - **Change**: forward_reward = self._forward_reward_weight * (x_velocity + 0.5 * x_velocity**2).
   - **Why**: A linear reward (x_velocity) provides a constant incentive per unit of velocity, which may not sufficiently distinguish between walking and running. The quadratic term (0.5 * x_velocity**2) exponentially rewards higher velocities, strongly incentivizing running speeds.
   - **Effect**: The RL agent will prioritize actions that maximize x_velocity, pushing the humanoid toward faster, running-like gaits.
3. **Added Posture Penalty**:
   - **Change**:
   - torso_angle = np.abs(self.data.qpos[3:7].sum())
   - posture_penalty = -10.0 * torso_angle if torso_angle > 0.3 else 0.0
   - **Why**: data.qpos[3:7] represents the quaternion orientation of the torso. Summing their absolute values approximates torso tilt. Penalizing large tilts (torso_angle > 0.3 radians) encourages an upright posture, which is natural for running and improves stability.
   - **Effect**: The humanoid avoids excessive leaning, maintaining a posture conducive to efficient running.
4. **Added Foot Clearance Reward**:
   - **Change**:
   - right_foot_z = self.data.xpios[5,2]
   - left_foot_z = self.data.xpios[8,2]
   - foot_clearance_reward = 2.0 * max(right_foot_z, left_foot_z) if max(right_foot_z, left_foot_z) > 0.2 else 0.0
   - **Why**: data.xpios[5,2] and data.xpios[8,2] are the z-coordinates of the right and left feet, respectively (based on body IDs for right_foot and left_foot). Rewarding high foot clearance (> 0.2 meters) encourages lifting feet during the swing phase, a key component of running to avoid tripping and ensure a smooth gait.
   - **Effect**: The humanoid learns to lift its feet higher, promoting a running gait with clear strides.
5. **Updated Reward Composition**:
   - **Change**: rewards = healthy_reward + forward_reward + foot_clearance_reward.
   - **Why**: Combining these rewards ensures the humanoid is incentivized for staying upright (healthy_reward), moving fast (forward_reward), and using a running gait (foot_clearance_reward).
   - **Effect**: The reward function holistically encourages running behavior.
6. **Maintained Costs**:
   - **Ctrl Cost**: ctrl_cost = self.control_cost(action), unchanged but uses the reduced ctrl_cost_weight (0.05).
   - **Contact Cost**: contact_cost = self.contact_cost, unchanged but potentially ineffective due to cfrc_ext issues.

    - **Why**: These costs penalize excessive actions and contact forces, but their reduced influence (via weights) ensures they don't hinder running.
    - **Effect**: Minimal penalties allow dynamic movements while discouraging wasteful actions.
7. **Updated reward_info**:
    - **Change**: Added "reward_posture": -posture_penalty to reward_info.
    - **Why**: Tracks the posture penalty for debugging and analysis, helping you monitor how much torso tilt affects the reward.
    - **Effect**: Provides visibility into the reward components during training, aiding in tuning.

Correction Needed:

- There's a typo in the code: right_foot_z = self.data.xpios[5,2] and left_foot_z = self.data.xpios[8,2].
    - **Issue**: xpios is likely a typo; it should be xipos, which stores the 3D positions of bodies in the MuJoCo model. xipos[5,2] and xipos[8,2] correspond to the z-coordinates of the right and left feet, respectively.
    - **Fix**:
    - right_foot_z = self.data.xipos[5,2]
    - left_foot_z = self.data.xipos[8,2]
    - **Why**: Correcting this ensures the foot clearance reward uses the actual z-coordinates of the feet.
    - **Action**: Update the file before training to avoid incorrect rewards.

Impact:

- The reward function now strongly incentivizes high forward velocity (linear and quadratic terms), upright posture, and high foot clearance, while penalizing excessive control and contact forces minimally.
- The humanoid is more likely to learn a running gait, as the rewards align with the biomechanics of running (fast movement, upright posture, clear strides).

2.3. step Method (Environment Dynamics)

Code Changes:

```
def step(self, action):

    xy_position_before = mass_center(self.model, self.data)

    self.do_simulation(action, self.frame_skip)

    xy_position_after = mass_center(self.model, self.data)



    xy_velocity = (xy_position_after - xy_position_before) / self.dt

    x_velocity, y_velocity = xy_velocity
```

```
    observation = self._get_obs()
    reward, reward_info = self._get_rew(x_velocity, action)

    # add fall penalty if terminated
    terminated = (not self.is_healthy) and self._terminate_when_unhealthy
    if terminated:
        reward += -200.0  # large penalty for falling
        reward_info["reward_fall"] = -200.0

    info = {
        "x_position": self.data.qpos[0],
        "y_position": self.data.qpos[1],
        "tendon_length": self.data.ten_length,
        "tendon_velocity": self.data.ten_velocity,
        "distance_from_origin": np.linalg.norm(self.data.qpos[0:2], ord=2),
        "x_velocity": x_velocity,
        "y_velocity": y_velocity,
        **reward_info,
    }

    if self.render_mode == "human":
        self.render()
    return observation, reward, terminated, False, info
```

Changes and Rationale:

1. **Added Fall Penalty**:

    - **Change**:
    - if terminated:
    - reward += -200.0
    - reward_info["reward_fall"] = -200.0
    - **Why**: Termination occurs when not self.is_healthy (torso z-coordinate outside [1.0, 2.0]) and terminate_when_unhealthy is True. A large negative reward (-200.0) strongly discourages falling, as falling ends the episode and prevents further rewards. This encourages the humanoid to maintain stability while running.
    - **Effect**: The RL agent learns to avoid actions that lead to falling, prioritizing gaits that keep the torso upright.
2. **Updated reward_info**:
    - **Change**: Added "reward_fall": -200.0 to reward_info when terminated.
    - **Why**: Tracks the fall penalty for debugging and analysis, allowing you to see when and why episodes end.
    - **Effect**: Helps diagnose if the humanoid is falling too often, indicating a need to adjust rewards or training.
3. **Unchanged Components**:
    - **Velocity Calculation**: xy_velocity = (xy_position_after - xy_position_before) / self.dt computes x_velocity, used in _get_rew.
    - **Observation**: observation = self._get_obs() provides the state (positions, velocities, etc.).
    - **Info Dictionary**: Includes x_position, y_position, tendon_length, x_velocity, etc., for monitoring.
    - **Why**: These components were correct and necessary for the environment's dynamics and RL training.
    - **Effect**: Ensures the environment provides accurate state and reward information.

Impact:

- The fall penalty reinforces stability, complementing the posture penalty and healthy_reward. The humanoid learns to run without falling, as falling incurs a significant penalty.

2.4. Other Methods (Unchanged)

- **Methods like reset_model, _get_obs, control_cost, contact_cost, is_healthy, _get_reset_info** were unchanged, as they were correct and functional.
- **Why**: These methods handle initialization, observation construction, cost calculations, and health checks, which were not the source of the running issue.
- **Effect**: The environment retains its core functionality, with changes focused on the reward structure.

Overall Impact of Code Changes:

- The modified environment now rewards:
    - High forward velocity (linear and quadratic terms).
    - Upright posture (posture penalty).
    - High foot clearance (foot clearance reward).
    - Stability (healthy reward, fall penalty).

- It allows dynamic actions (reduced ctrl_cost_weight) and minimizes irrelevant penalties (contact_cost).
- The humanoid is now set up to learn a fast, stable, and natural running gait, addressing the original issues.