helicopter_agent.m

Table of Contents

Summary: Helicopter control agent. Contains the functions needed to communicate with the RL-Glue Core.

Contents

1. `helicopter_agent`: set useful paths and function handles
2. `helicopter_agent_init`: initialize data structure
3. `helicopter_agent_start`: take the first step
4. `helicopter_agent_step`: take a step
5. `helicopter_agent_message`: communicate between trainer and agent
6. `helicopter_agent_end`: finish episode
7. `helicopter_agent_cleanup`: save data structure

Code

```
function theAgent=helicopter_agent(policy_input, codec_base, pilco_root)
% Add paths and fill agent structure

    global policy
    policy = policy_input;

    addpath([codec_base 'agent'], [codec_base 'glue'], codec_base);

    addpath([pilco_root 'base'],[pilco_root 'util'],[pilco_root 'gp'],[pilco_root

    theAgent.agent_init=@helicopter_agent_init;
    theAgent.agent_start=@helicopter_agent_start;
    theAgent.agent_step=@helicopter_agent_step;
    theAgent.agent_end=@helicopter_agent_end;
    theAgent.agent_cleanup=@helicopter_agent_cleanup;
    theAgent.agent_message=@helicopter_agent_message;

end

function helicopter_agent_init(taskSpec)
% This is a persistent struct we will use to store the data collected for
% the next learning iteration
global helicopter_agent_struct;
helicopter_agent_struct = zeros(2,17);
```

```
end

function theAction=helicopter_agent_start(theObservation)
% Take the first step of the agent as the episode starts, and store data

    global helicopter_agent_struct;
    global timeStep;
    global policy
    timeStep = 1;

    theAction = org.rlcommunity.rlglue.codec.types.Action();
 theAction.doubleArray = policy.fcn(policy, theObservation.doubleArray, zeros(leng

    helicopter_agent_struct(timeStep, 1:12) = theObservation.doubleArray;
    helicopter_agent_struct(timeStep, 13:16) = theAction.doubleArray;
    helicopter_agent_struct(timeStep, 17) = 0;

end

function theAction=helicopter_agent_step(theReward, theObservation)
% Take a step and store data

    global helicopter_agent_struct;
    global timeStep;
    global policy;

    theAction = org.rlcommunity.rlglue.codec.types.Action();
    theAction.doubleArray = policy.fcn(policy, theObservation.doubleArray, zeros(l

    timeStep = timeStep + 1;
    helicopter_agent_struct(timeStep, 1:12) = theObservation.doubleArray;
    helicopter_agent_struct(timeStep, 13:16) = theAction.doubleArray;
    helicopter_agent_struct(timeStep, 17) = theReward;

end

function helicopter_agent_end(theReward)
    % An episode ends
end

function returnMessage=helicopter_agent_message(theMessageJavaObject)
% Custom function for trainer-agent communication

    inMessage=char(theMessageJavaObject);
    global policy;

    if strcmp(inMessage,'what is your name?')==1
  returnMessage='my name is helicopter_agent, Matlab edition!';
    elseif strcmp(inMessage, 'when')==1
        % Print policy training timestamp
        returnMessage = num2str(policy.date);
    else
        returnMessage='I don\''t know how to respond to your message';
    end

end

function helicopter_agent_cleanup()
% On cleanup, save the collected data to a MAT-file

    global helicopter_agent_struct;
    save('GPHistory', 'helicopter_agent_struct');
end
```

Published with MATLAB® R2014a