RL agent Donor agnostic (RL only)

DQN without recurring neural network

format shortG

"Randomization"

75/25 - 8/4 training/validation split

allmodelfoldincs = readmatrix("allmodelfoldincs.txt")

modelinitcons = readmatrix("modelinitcons.txt")

rawfoldincs = readmatrix("rawfoldincs.txt")

rawinitcons = readmatrix("rawinitcons.txt")

donoridxmap = {1,2,3,4,5:6,7,8,9:11,12:14,15:27,28:32,33:37}

Run training and validation per seed

for seed=0:4

seed

[trialfoldincs, trialinitcons] = initializedatasets(seed,allmodelfoldincs,modelinitcons,rawfoldincs,rawinitcons,donoridxmap)

dqnagent = dqninittraining(seed);

[simulationPlots,simulationResults] = simwithset(trialfoldincs,trialinitcons,dqnagent);

[rewardMatrix,totalRewards,totalReward,averageReward] = calculatescore(6,simulationResults)

end

function [trialfoldincs, trialinitcons] = initializedatasets(seed,allmodelfoldincs,modelinitcons,rawfoldincs,rawinitcons,donoridxmap)

rng(seed);

trainingvector = sort(randperm(12,8),"ascend")

validationvector = [];

for i=1:12

if ismember(i,trainingvector)==false

validationvector = [validationvector i];

end

end

validationvector = sort(validationvector,"ascend")

trialmodelfoldincs = [];

for i=trainingvector

trialmodelfoldincs = [trialmodelfoldincs allmodelfoldincs(:,(numel(modelinitcons)\*(i-1)+1):(numel(modelinitcons)\*(i-1)+numel(modelinitcons)))];

end

trialmodelfoldincs

writematrix(trialmodelfoldincs, "modelfoldincs.txt")

trialfoldincs = [];

trialinitcons = [];

for i=validationvector

trialfoldincs = [trialfoldincs rawfoldincs(:,donoridxmap{i})];

trialinitcons = [trialinitcons rawinitcons(donoridxmap{i})];

% POTENTIALLY BETTER IF ALL NORMALIZED TO 500000

end

end

function dqnagent = dqninittraining(seed)

rng(seed);

% create first environment

env = modelEnv241;

env.useCustom = false;

obs = getObservationInfo(env);

acts = getActionInfo(env);

validateEnvironment(env)

% hyperparameters

discountFactor = 1;

epsilon = 1;

initOpts = rlAgentInitializationOptions;

initOpts.NumHiddenUnit = 64;

initOpts.UseRNN = false;

dqnagentoptions = rlDQNAgentOptions;

dqnagentoptions.DiscountFactor = discountFactor;

dqnagentoptions.EpsilonGreedyExploration.Epsilon = epsilon;

dqnagentoptions.SequenceLength = 1;

dqnagentoptions.UseDoubleDQN = false;

dqnagent = rlDQNAgent(obs,acts,initOpts,dqnagentoptions);

% plot(layerGraph(getModel(getCritic(dqnagent)).Layers));

trainopts = rlTrainingOptions;

trainopts.MaxEpisodes = 1500;

trainopts.StopTrainingCriteria = "EpisodeCount";

trainopts.StopTrainingValue = 300;

% trainopts.StopTrainingCriteria = "AverageReward";

% trainopts.StopTrainingValue = 8;

%trainopts.Plots = 'none';

%trainopts.Verbose = true;

plot(env)

results = train(dqnagent,env,trainopts);

%save agent

save(append("agent\_seed\_",string(seed)),"dqnagent")

end

function [plots,experiences] = simwithset(foldExMaps,day3cons,sarsaagent3)

plots = cell(size(foldExMaps,2),1);

experiences = cell(size(foldExMaps,2),1);

for i=1:size(foldExMaps,2)

%creating foldexmap

foldExMap1 = foldExMaps(:,i);

foldExMap1 = foldExMap1(~isnan(foldExMap1));

idx = foldExMap1==1;

idx(1) = false;

foldExMap1(idx) = [];

%creating new env

env3 = modelEnv241;

env3.useCustom = true;

env3.foldExMap = foldExMap1;

%NORMALIZED INIT CONS

%

env3.initialState = [100000/1000000,1];

validateEnvironment(env3)

% plot now

plot(env3);

experiences{i} = sim(sarsaagent3,env3);

plots{i} = env3.Figure;

end

end

function [rewardMatrix,totalRewards,totalReward,averageReward] = calculatescore(maxSeriesLength,simulationResults)

maxSeriesLength = 5;

rewardMatrix = [0:(maxSeriesLength-1)]';

for i=1:size(simulationResults,1)

seriesRewards = simulationResults{i}.Reward.Data;

if size(seriesRewards,1)<maxSeriesLength

numZeros = maxSeriesLength - size(seriesRewards,1);

for j=1:numZeros

seriesRewards = [seriesRewards; 0];

end

end

rewardMatrix = [rewardMatrix seriesRewards];

end

totalRewards = [];

for i=2:size(rewardMatrix,2)

totalRewards = [totalRewards sum(rewardMatrix(:,i))];

end

totalReward = sum(totalRewards);

averageReward = mean(totalRewards);

end