RL agent Donor agnostic no penalty no u donors

DQN without recurring neural network

format shortG

seed = 4

epsilondecay = 0.0003125;

numepisodes = 2000;

learnrate = 0.01;

hiddenlayers = 64;

batchsize = 64;

agentnumber = 18;

No U donors

allmodelfoldincs = readmatrix("allmodelfoldincs.txt")

modelinitcons = readmatrix("modelinitcons.txt")

rawfoldincs = readmatrix("rawfoldincs.txt")

rawinitcons = readmatrix("rawinitcons.txt")

truecon = readtable("true con.xlsx");

truecon(:,1)=[];

rawdonornames = truecon.Properties.VariableNames;

rawdonornames = string(rawdonornames);

rawdonornames = strrep(rawdonornames,"\_","-")

% rawdonornames = ["T054","T046","T051","T062","T031","T031","T052","T038","T036","T036","T036","T066","T066","T066","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA4","UA141","UA141","UA141","UA141","UA141","UA6","UA6","UA6","UA6","UA6"]

donorlist = ["T054","T046","T051","T062","T031","T052","T038","T036","T066","UA4","UA141","UA6"]

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

Run training and validation

Randomization:

-6:3 training:validation split for tall the T donors

u donors never included

numdonors = 6

All validation set printed

initializedatasets(seed,allmodelfoldincs,modelinitcons,rawfoldincs,rawinitcons,rawdonornames,donoridxmap,donorlist);

numdonors = 6;

dqnagent = dqninittraining(seed,numdonors,epsilondecay,numepisodes,hiddenlayers,batchsize,learnrate);

save("Hyperparameter Tuning/DQN 2022/agent\_248\_"+agentnumber,"dqnagent")

rawfoldincs(:,15:end) = []

rawinitcons(15:end) = []

% dqnagent = load("agent\_248\_hailmary"+seed,"dqnagent").dqnagent

[simulationPlots,simulationResults] = simwithset(rawfoldincs,rawinitcons,rawdonornames,dqnagent);

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

writematrix(rewardMatrix,"Hyperparameter Tuning/DQN 2022/rewardMatrix\_248\_"+agentnumber+".xlsx")

writematrix(totalRewards,"Hyperparameter Tuning/DQN 2022/totalRewards\_248\_"+agentnumber+".xlsx")

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

rng(seed);

trainingvector = sort(randperm(9,6),"ascend");

trainingset = donorlist(trainingvector)

validationvector = [];

for i=1:12

if ismember(i,trainingvector)==false

validationvector = [validationvector i];

end

end

validationvector = sort(validationvector,"ascend");

validationset = donorlist(validationvector)

availmodelfoldincs = [allmodelfoldincs(:,1:numel(modelinitcons)\*9)];

trialmodelfoldincs = [];

for i=trainingvector

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

end

writematrix(trialmodelfoldincs, "modelfoldincs.txt")

% trialfoldincs = [];

% trialinitcons = [];

% trialdonornames = [];

% for i=validationvector

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

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

% % POTENTIALLY BETTER IF ALL NORMALIZED TO 500000

% trialdonornames = [trialdonornames rawdonornames(donoridxmap{i})];

% end

end

function dqnagent = dqninittraining(seed,numdonors,epsilondecay,numepisodes,hiddenlayers,batchsize,learnrate)

rng(0);

% create first environment

env = modelEnv241;

env.useCustom = false;

env.numdonors = numdonors;

obs = getObservationInfo(env);

acts = getActionInfo(env);

validateEnvironment(env)

% hyperparameters

discountFactor = 1;

epsilon = 1;

initOpts = rlAgentInitializationOptions;

initOpts.NumHiddenUnit = hiddenlayers;

initOpts.UseRNN = false;

criticOpts = rlOptimizerOptions('LearnRate',learnrate);

dqnagentoptions = rlDQNAgentOptions;

dqnagentoptions.DiscountFactor = discountFactor;

dqnagentoptions.EpsilonGreedyExploration.Epsilon = epsilon;

dqnagentoptions.EpsilonGreedyExploration.EpsilonDecay = epsilondecay;

dqnagentoptions.SequenceLength = 1;

dqnagentoptions.UseDoubleDQN = false;

dqnagentoptions.MiniBatchSize = batchsize;

dqnagentoptions.CriticOptimizerOptions = criticOpts;

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

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

trainopts = rlTrainingOptions;

trainopts.MaxEpisodes = numepisodes;

trainopts.StopTrainingCriteria = "EpisodeCount";

trainopts.StopTrainingValue = numepisodes;

% trainopts.StopTrainingCriteria = "AverageReward";

% trainopts.StopTrainingValue = 15;

%trainopts.Plots = 'none';

%trainopts.Verbose = true;

plot(env)

results = train(dqnagent,env,trainopts);

end

function [plots,experiences] = simwithset(foldExMaps,day3cons,donornames,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];

env3.modelInfo = donornames(i)

validateEnvironment(env3)

% plot now

plot(env3);

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

plots{i} = env3.Figure;

end

end

function results3 = trainwithset(foldExMaps,day3cons,sarsaagent3,trainopts1)

results3 = cell(size(foldExMaps,2),1);

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

for i=1:size(foldExMaps,2)

%creating foldexmap

foldExMap1 = foldExMaps(:,i);

idx = foldExMap1==1;

idx(1) = false;

foldExMap1(idx) = [];

%creating new env

env3 = modelEnv10;

env3.useCustom = true;

env3.foldExMap = foldExMap1;

env3.initialState = [day3cons(i),1];

validateEnvironment(env3)

%train

results3{i} = train(sarsaagent3,env3,trainopts1);

plot(env3);

sim(sarsaagent3,env3);

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