Setup 2: Discrete range rewards, 2 food items

May 17, 2020

[54]:
```
#Experiments to run: Exp 1 -- random initialization of agent and goal positions␣
 ↪for different environments and compute fitness
#In static environment -- having more rewards may not necessarily be better
#Exp 2 -- same experiment but now the different goals will disappear after␣
 ↪certain time-steps
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
import time
import random
import numpy as np
import pandas as pd
import seaborn as sns

import os
import sys
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path)

from pyrlap.domains.gridworld import GridWorld
from pyrlap.algorithms.qlearning import Qlearning
from pyrlap.domains.gridworld.gridworldvis import visualize_trajectory
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# define the gridworld here

def create_array():
    y_loc = (random.randint(1, 3), 2) #rewards only in 2nd half of grid world,␣
 ↪in different columns
    #x_loc = (random.randint(1, 3), 5)
    #z_loc = (random.randint(1, 3), 8)
    #q_loc = (random.randint(6, 8), 2) #rewards only in 2nd half of grid world,␣
 ↪in different columns
    #w_loc = (random.randint(6, 8), 5)
    e_loc = (random.randint(6, 8), 8)
    locs = [y_loc, e_loc]

    gw_array = ['..........', '..........', '..........', '..........', '........
↪..',
               '..........', '..........', '..........', '..........', '........
↪..',
               ]
    #n*n grid

    s = gw_array[y_loc[0]]
    new = list(s)
    new[y_loc[1]] = 'y'
    s = ''.join(new)
    gw_array[y_loc[0]] = s

    s = gw_array[e_loc[0]]
    new = list(s)
    new[e_loc[1]] = 'e'
    s = ''.join(new)
    gw_array[e_loc[0]] = s



    return gw_array, (locs)

'''

    s = gw_array[x_loc[0]]
    new = list(s)
    new[x_loc[1]] = 'x'
    s = ''.join(new)
    gw_array[x_loc[0]] = s

    s = gw_array[q_loc[0]]
    new = list(s)
    new[q_loc[1]] = 'q'
    s = ''.join(new)
    gw_array[q_loc[0]] = s

    s = gw_array[z_loc[0]]
    new = list(s)
    new[z_loc[1]] = 'z'
    s = ''.join(new)
    gw_array[z_loc[0]] = s
    s = gw_array[w_loc[0]]
    new = list(s)
    new[w_loc[1]] = 'w'
    s = ''.join(new)
    gw_array[w_loc[0]] = s


'''



def create_gridworld(rewards={'.':0, 'y':0, 'e':0}):
    gw_array, locs = create_array() #get the array and the location of goals to␣
 ↪compute fitness
    in_state = (random.randint(4, 6), random.randint(4, 6)) #initial location at␣
 ↪bottom left corner
    gw = GridWorld(gridworld_array=gw_array,
    init_state=in_state, #random initialization of the initial state
    feature_rewards=rewards) #the reward function
    return gw, locs

def learn(gw,m):
    np.random.seed(1234)
    all_run_data = []
    for i in range(1):
        params = {'learning_rate': 1,
              'eligibility_trace_decay': .8,
              'initial_qvalue': 10}
        qlearn = Qlearning(gw,
                      softmax_temp=.2,
                      discount_rate=.99,
                      **params)
        run_data = qlearn.train(episodes=1,
                           max_steps=m,
                           run_id=i,
                           return_run_data=True)
        for r in run_data:
            r.update(params)
        all_run_data.extend(run_data)

    return qlearn, (all_run_data)

def plot_learnt(qlearn, gw): #function to plot the learnt trajectory
    traj = qlearn.run(softmax_temp=0.0, randchoose=0.0, max_steps=1000)
    gwp = gw.plot()
    gwp.plot_trajectory(traj=[(s, a) for s, a, ns, r in traj])

def plot_history(gw, all_run_data):
    run_df = pd.DataFrame(all_run_data) #convert to pandas
    state = run_df["s"] #get states visited
    state_array = state.values #convert to matrix

    action = run_df["a"] #get actions taken
    action_array = action.values #convert to matrix

    gwp = gw.plot() #plot it
    gwp.plot_trajectory(traj=[(s, a) for s in state_array for a in action_array])

def compute_fitness(all_run_data, locs, gw): #function computes fitness over the␣
 ↪history (you can also compute fitness over the learnt policy)
    fitness = 0

    run_df = pd.DataFrame(all_run_data) #convert to pandas
    state = run_df["s"] #get states visited
    state_array = state.values #convert to matrix

    for i in range(len(state_array)):
        a = state_array[i]
        if((9-a[1],a[0]) in locs): #loop through the traj, if traj position is␣
 ↪not same as food position then fitness decreases, else increases by one
            fitness = fitness+1
        else:
            fitness = fitness-0.1
    return fitness
```

[4]:
```
## REWARDS = 3, Discrete range, all combinations, THOUSAND steps, CENTER initial␣
 ↪state
# Initial state = center
# Episodes = 50
# Trials = 10
# Steps = 1000
# Environments = 10

fitness_vals_thousand = []
y_vals_thousand = []
e_vals_thousand = []

for y in range(-3, 4):
    for e in range(-3, 4):
        rewards={'.':0, 'y':y*10, 'e':e*10}
        fit = []
        for env in range(10): #Repeatedly sample lots of environments
            gw, locs = create_gridworld(rewards) #create gridworld here
            qlearn, all_run_data = learn(gw, 10000)     #Q-learn here
            fitness= compute_fitness(all_run_data, locs, gw) #compute fitness␣
↪here for the enviroment
            fit.append(fitness)
            if (env == 1):
                print("Rewards are set as:")
                print('y =', y, 'e =', e)
                plot_learnt(qlearn, gw)


        fitness_vals_thousand.append(sum(fit)/float(len(fit)))
        y_vals_thousand.append(y)
        e_vals_thousand.append(e)
        print("Average Fitness", sum(fit)/float(len(fit)))
```

```
Rewards are set as:
y = -3 e = -3
Average Fitness -953.2500000001543
Rewards are set as:
y = -3 e = -2
Average Fitness -958.9700000001545
Rewards are set as:
y = -3 e = -1
Average Fitness -955.1200000001545
Rewards are set as:
y = -3 e = 0
Average Fitness -856.2300000001445
Rewards are set as:
y = -3 e = 1
Average Fitness 3390.760000000327
Rewards are set as:
y = -3 e = 2
Average Fitness 3324.540000000316
Rewards are set as:
y = -3 e = 3
Average Fitness 3376.5700000003294
Rewards are set as:
y = -2 e = -3
Average Fitness -959.9600000001541
Rewards are set as:
y = -2 e = -2
Average Fitness -956.4400000001548
Rewards are set as:
y = -2 e = -1
Average Fitness -957.4300000001547
Rewards are set as:
y = -2 e = 0
Average Fitness -848.530000000145
```

```
Rewards are set as:
y = -2 e = 1
Average Fitness 3357.4300000003254
Rewards are set as:
y = -2 e = 2
Average Fitness 3307.2700000003097
Rewards are set as:
y = -2 e = 3
Average Fitness 3316.9500000003195
Rewards are set as:
y = -1 e = -3
Average Fitness -955.6700000001543
Rewards are set as:
y = -1 e = -2
Average Fitness -958.200000000155
Rewards are set as:
y = -1 e = -1
Average Fitness -958.9700000001546
Rewards are set as:
y = -1 e = 0
Average Fitness -845.1200000001439
Rewards are set as:
y = -1 e = 1
Average Fitness 3295.7200000003186
Rewards are set as:
y = -1 e = 2
Average Fitness 3355.3400000003194
Rewards are set as:
y = -1 e = 3
Average Fitness 3390.430000000323
Rewards are set as:
y = 0 e = -3
Average Fitness -845.2300000001436
Rewards are set as:
y = 0 e = -2
Average Fitness -847.1000000001432
Rewards are set as:
y = 0 e = -1
Average Fitness -850.0700000001436
Rewards are set as:
y = 0 e = 0
Average Fitness -747.7700000001317
Rewards are set as:
y = 0 e = 1
Average Fitness 3292.2000000003122
Rewards are set as:
y = 0 e = 2
Average Fitness 3355.010000000326
```

```
Rewards are set as:
y = 0 e = 3
Average Fitness 3181.6500000003
Rewards are set as:
y = 1 e = -3
Average Fitness 3335.320000000325
Rewards are set as:
y = 1 e = -2
Average Fitness 3349.8400000003203
Rewards are set as:
y = 1 e = -1
Average Fitness 3349.730000000328
Rewards are set as:
y = 1 e = 0
Average Fitness 3377.7800000003285
Rewards are set as:
y = 1 e = 1
Average Fitness 3406.380000000335
Rewards are set as:
y = 1 e = 2
Average Fitness 3371.07000000033
Rewards are set as:
y = 1 e = 3
Average Fitness 3375.2500000003297
Rewards are set as:
y = 2 e = -3
Average Fitness 3322.450000000321
Rewards are set as:
y = 2 e = -2
Average Fitness 3346.760000000331
Rewards are set as:
y = 2 e = -1
Average Fitness 3333.2300000003256
Rewards are set as:
y = 2 e = 0
Average Fitness 3302.210000000324
Rewards are set as:
y = 2 e = 1
Average Fitness 3421.2300000003343
Rewards are set as:
y = 2 e = 2
Average Fitness 3400.2200000003286
Rewards are set as:
y = 2 e = 3
Average Fitness 3366.3400000003276
Rewards are set as:
y = 3 e = -3
Average Fitness 3379.760000000332
```

```
Rewards are set as:
y = 3 e = -2
Average Fitness 3327.6200000003228
Rewards are set as:
y = 3 e = -1
Average Fitness 3353.9100000003227
Rewards are set as:
y = 3 e = 0
Average Fitness 3399.0100000003317
Rewards are set as:
y = 3 e = 1
Average Fitness 3415.7300000003315
Rewards are set as:
y = 3 e = 2
Average Fitness 3362.160000000332
Rewards are set as:
y = 3 e = 3
Average Fitness 3394.8300000003283
```