Crowdworker-Behavior_Tutorial

September 22, 2022

1 A Model of Adaptive Crowdworker Behavior

Tutorial on the use of the crowdworker reinforcement learning environment we provide as Supplementary Material to the paper.

1.1 Installation

If not already done, you need to install Python 3.8 and the required libraries. You can do this with conda via:

```
conda create -n crowdworker_env python=3.8
conda activate crowdworker_env
pip install -r requirements.txt
```

Now we can start.

1.2 Running a Trained Model

We will start with an already trained model and its corresponding crowdworking environment to explore the first steps. We will start with loading the agent/crowdworker model and its environment and settings.

[11]:
```
# some imports
import os
from task import TaskPropertiesDistribution, AntiCheatSettings
from util.exputil import Config
from user import UserProperties
from userenv import UserModelEnv
import sb3_contrib

# name of the configuration file
# this is for a model that was trained in an environment with cheating␣
 ↪deterrents
# the model is stored in ../exp/[name]
name = "paper-exps_cheating_qa3-0.1_rep0.9"

# if you like, you can change this to
```

```
#name = "paper-exps_cheating_qa3-0_rep0.9"
# if you want to analyze a worker agent in an environment without cheating␣
 ↪deterrents

# let's load this experimental configuration (just a json file, you can␣
 ↪actually take a look into it, if you like)
config = Config.load(name)

# now we load the worker and the environment

# properties of the worker, e.g. how much he or she values monetary payout and␣
 ↪interestigness
user_props = UserProperties.load(config)

# properties of the task-givers (here, we have 5 task-givers)
task_prop_distributions = TaskPropertiesDistribution.load_list(config)

# settings of the cheating deterrents
anti_cheat_settings = AntiCheatSettings.load(config)

# create the RL environment, which corresponds to the crowdworking platform
env = UserModelEnv(config, user_props, task_prop_distributions,␣
 ↪anti_cheat_settings)

# loading the actual RL model (machine learning model)
assert config.rl_model == "QR-DQN"
model = sb3_contrib.qrdqn.QRDQN.load(config.path("model.save"), env=env)
```

Now that we have loaded everything, we can let the agent work on the crowdworking platform and check what it does.

We will see that the worker selects a task, works on it diligently and once it runs out of questions for the task, it switches to the next task.

(If you like, you can change above the pretrained model to the one that was trained in an environment without cheating deterrents and therefore acts negligently most of the time. Just load the other model above. Note that then obviously the results change!)

[12]:
```
# setting a seed to make results reproducible
env.seed(98765)

obs = env.reset() # a new episode
while True:
    # let the RL model analyze the observation (what the worker currently sees)
    # and decide what to do (predict an action)
    action, _states = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
```

```
    # action is an index, convert to a human-readable string representation
    action_as_str = UserModelEnv.action_to_str(action)
    print(f"Worker performed action {action_as_str} and earned a reward of␣
 ↪{reward:.2f}")

    if done: # episode has ended because worker has run out of time or has quit
        break
```

```
Worker performed action SWITCH TO TASK 3 and earned a reward of 0.00
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
```

```
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.59
Worker performed action SWITCH TO TASK 4 and earned a reward of 0.00
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
```

```
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.76
Worker performed action ANSWER DILIGENTLY and earned a reward of 0.00
```

If we want to take a closer look at what the agent can see, we can print out the observations.

We can see here, that at the beginning of the episode, the agent can not see how interesting a task is or how much time effort it requires. Only the payout is visible.

Once the agent tries out a task, it finds out how interesting it is and how much effort it takes.

[13]:
```
# setting a seed to make results reproducible
env.seed(98765)

# a new episode
obs = env.reset()

print(f"At beginning of the episode:\n {env.observation_to_string(obs)}")

# let's take some manual steps in the environment (instead of the trained RL␣
 ↪model)
env.step(UserModelEnv.SWITCH_TASK0+1) # select task 1
obs, reward, done, info  = env.step(UserModelEnv.ACTION_ANS_INTENT) # answer␣
 ↪diligently

print(f"After trying out Task 1:\n {env.observation_to_string(obs)}")
```

```
At beginning of the episode:
 Observation:
  Task 0:
      payout 0.6253242330848118 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 1:
      payout 0.33753537837799463 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 2:
      payout 0.5568837901272194 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 3:
      payout 0.6333947080897941 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 4:
      payout 0.6768648101640447 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  current task: -1.0
```

```
  reputation: 1.0
  time: 0.0/50.0

After trying out Task 1:
 Observation:
  Task 0:
      payout 0.6253242330848118 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 1:
      payout 0.33753537837799463 | rounds 1.0
      expert 0.8589561657442589 | effort 0.5247141825974938 | interest
0.037395870193654535
  Task 2:
      payout 0.5568837901272194 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 3:
      payout 0.6333947080897941 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  Task 4:
      payout 0.6768648101640447 | rounds 0.0
      expert -1.0 | effort -1.0 | interest -1.0
  current task: 1.0
  reputation: 1.0
  time: 1.524714182597494/50.0
```

We can inspect the behavior of the agent more in depth. Here, we analyze for which task-giver the agent works. Since the properties of the tasks are beta-distributed (so in each episode they are somewhat different), we repeat the experiment multiple times and take the average.

[14]:
```python
# imports
import numpy as np

# setting a seed to make results reproducible
env.seed(98765)

# 100 episodes or repetitions
# (in the paper, we always use 1000 repetitions, but that takes a bit more time)
repetitions = 100

# counting how many questions answered in total and for task_giver_0
# list of counters for all repetitions
# we use lists so that we can compute the mean over all episodes
counters_for_all_task_givers = []
counters_for_task_giver_0 = []

for i in range(repetitions):
    # a new episode
```

```
    obs = env.reset()

    # reset counters for this episode
    counter_for_all_task_givers = 0
    counter_for_task_giver_0 = 0

    while True:
        # let the RL model analyze the observation (what the worker currently␣
↪sees)
        # and decide what to do (predict an action)
        action, _states = model.predict(obs, deterministic=True)
        obs, reward, done, info = env.step(action)

        # if the worker answered a question / the agent did an answering action
        if action == UserModelEnv.ACTION_ANS_INTENT or action == UserModelEnv.
↪ACTION_ANS_RND:
            counter_for_all_task_givers += 1

            if env.current_task_idx != -1: # should not happen, but just in␣
↪case the agent tries to answer a question before selecting a task

                # the list of tasks is randomized at the beginning of every␣
↪episode so that the agent
                # can not learn a connection between task-givers and tasks (for␣
↪the case that task-givers
                # have different distributions). With this map, we can get the␣
↪task-giver for a task.
                task_giver_for_current_task = env.task_task_dist_map[env.
↪current_task_idx]

                if task_giver_for_current_task == 0:
                    counter_for_task_giver_0 += 1

        if done: # episode has ended because worker has run out of time or has␣
↪quit

            # add the measurements to the list of counters
            counters_for_all_task_givers.append(counter_for_all_task_givers)
            counters_for_task_giver_0.append(counter_for_task_giver_0)

            break

ratio_task_giver_0 = np.mean([i/j for i,j in zip(counters_for_task_giver_0,␣
↪counters_for_all_task_givers)])
print(f"The agent answered on average {np.mean(counters_for_all_task_givers)}␣
↪questions per episode.")
```

```
print(f"It answered on average {np.mean(counters_for_task_giver_0)} questions␣
 ↪for task-giver 0.")
print(f"This means, on average {ratio_task_giver_0} of the questions were for␣
 ↪task-giver 0.")
print(f"In the long run, with 5 task-givers, one would expect a ratio of {1/5}␣
 ↪questions.")
```

```
The agent answered on average 85.39 questions per episode.
It answered on average 17.46 questions for task-giver 0.
This means, on average 0.21880928415930798 of the questions were for task-giver
0.
In the long run, with 5 task-givers, one would expect a ratio of 0.2 questions.
```

1.3 Changing the Environment and Analyzing Effects

We can change the crowdworking environment and see how this affects the behavior of the worker agent.

If you change the environment substantionally, you will have to retrain the RL model, so that it can learn how to work with these changes. We will see how to do this retraining below. Here, however, we only do a small change, so we can still use the pre-trained environment.

We will create a new environment where task-giver 0 has a higher payout. In the paper, we saw that an increased payout motivates the worker to work more on tasks by this task-giver. We see the same effect replicated here. For more information on the effect, take a look at the paper, Section 5, effect A.1.

[15]:
```
# Very similar code to above. The only change is the modified␣
 ↪task_prop_distributions
# (i.e. the properties of the task-givers)

# preparing the environment and loading the RL model
from task import TaskPropertiesCustomFixedDistribution,␣
 ↪TaskPropertiesCustomBetaDistribution, TaskPropertiesBetaDistribution

name = "paper-exps_cheating_qa3-0.1_rep0.9"
config = Config.load(name)
user_props = UserProperties.load(config)

# properties of the task-givers
# instead of loading task-property-distributions (task-givers),
# here we create new ones. The first task-giver (task-giver 0) has a higher␣
 ↪payout.
# you can take a look at task.py to see how you can change the task-giver␣
 ↪distributions.
# you can also use TaskPropertiesCustomFixedDistribution if you want fixed␣
 ↪values instead of distributions.
```

```
# task-giver 0 has a shifted beta distribution, the other task-givers stay the same.
task_prop_distributions = [TaskPropertiesCustomBetaDistribution(payout=(90, 10)),
                           TaskPropertiesBetaDistribution(),
                           TaskPropertiesBetaDistribution(),
                           TaskPropertiesBetaDistribution(),
                           TaskPropertiesBetaDistribution()]

# the other settings are loaded again
anti_cheat_settings = AntiCheatSettings.load(config)
env = UserModelEnv(config, user_props, task_prop_distributions, anti_cheat_settings)
assert config.rl_model == "QR-DQN"
model = sb3_contrib.qrdqn.QRDQN.load(config.path("model.save"), env=env)

# now we run the simulation and analyze how the agent reacts
# the rest of the code is identical to the commented version above

env.seed(98765)
repetitions = 100
counters_for_all_task_givers = []
counters_for_task_giver_0 = []

for i in range(repetitions):
    obs = env.reset()

    counter_for_all_task_givers = 0
    counter_for_task_giver_0 = 0

    while True:
        action, _states = model.predict(obs, deterministic=True)
        obs, reward, done, info = env.step(action)

        if action == UserModelEnv.ACTION_ANS_INTENT or action == UserModelEnv.ACTION_ANS_RND:
            counter_for_all_task_givers += 1

            if env.current_task_idx != -1:
                task_giver_for_current_task = env.task_task_dist_map[env.current_task_idx]

                if task_giver_for_current_task == 0:
                    counter_for_task_giver_0 += 1

        if done:
```

```
            counters_for_all_task_givers.append(counter_for_all_task_givers)
            counters_for_task_giver_0.append(counter_for_task_giver_0)

            break

ratio_task_giver_0 = np.mean([i/j for i,j in zip(counters_for_task_giver_0,␣
 ↪counters_for_all_task_givers)])
print(f"The agent answered on on average {ratio_task_giver_0} of the questions␣
 ↪for task-giver 0.")
print(f"This is much higher than the ratio we saw above when all task-givers␣
 ↪were equal.")
```

```
The agent answered on on average 0.5027607153753726 of the questions for task-
giver 0.
This is much higher than the ratio we saw above when all task-givers were equal.
```

1.4 Training a New Model

If you change the crowdworking environment in a larger way, you need to train a new RL model so that it can learn what the optimal behavior in this new setting is. Here, we show how to perform the training.

While the RL model can be trained on a CPU, the training of the deep neural network is much faster if you have a GPU. For this tutorial, we only train the RL model for a few steps so that you can see how the code for the training works. You can of course perform a long training and evaluate your own trained model.

[17]:
```
# imports
from rllearning import rl_training

# the name of your setting, a directory will be created in ../exp/[name]
name = f"tutorial_model_no-cheating-deterrent"

config = Config.create(name, exist_ok=True)
config.description = "A model build in the tutorial without cheating deterrents.
 ↪"

# properties of the worker (how much they care for the interestingness and the␣
 ↪payout reward,
# how large their time budget is and the reputation level they start with)
user_props = UserProperties(interestingness_sensitivity=0.5,␣
 ↪payout_sensitivity=1,
                            time_sensitivity=0, time_budget=50,␣
 ↪start_reputation=1)

# default task-giver property distributions
task_prop_distributions = [TaskPropertiesBetaDistribution()] * 5
```

```
# cheating deterrents
ban_after = 3 # number of gold questions answered incorrectly after which the␣
 ↪worker is banned from the task
qa_prct = 0 # probability of a hidden gold question
reputation_delta = 0.05 # how the reputation changes with a correct or␣
 ↪incorrect gold question
reputation_lvl = 0 # minimum reputation level
anti_cheat_settings = AntiCheatSettings(ban_after, qa_prct, -reputation_delta,␣
 ↪+reputation_delta, reputation_lvl)

# hyperparemters of the deep RL training
config.rl_model = "QR-DQN"
config.exploration_fraction = 0.2
config.exploration_final_eps = 0.05
config.learning_starts = 500

# the number of training steps
config.total_timesteps = 10
# for the tutorial, set to 10, just to see that it works
# if you want to actually train the model, use
#config.total_timesteps = 10000000

# seed for reproducable trainings
config.main_seed = 12345

# save all the settings everything
config.save()
user_props.save(config)
TaskPropertiesBetaDistribution.save_list(task_prop_distributions, config)
anti_cheat_settings.save(config)

rl_training(config, user_props, task_prop_distributions, anti_cheat_settings)
print(f"This was not a full training as we only trained for {config.
 ↪total_timesteps} steps!")
print("Increase config.total_timesteps for an actual training.")

print(f"You can load this model by using the name '{config.name}'")
```

```
{'name': 'tutorial_model_no-cheating-deterrent', 'timestamp': '20:15PM CEST on
Sep 20, 2022', 'exp_dir_path': '../exp/tutorial_model_no-cheating-deterrent',
'description': 'A model build in the tutorial without cheating deterrents.',
'rl_model': 'QR-DQN', 'exploration_fraction': 0.2, 'exploration_final_eps':
0.05, 'learning_starts': 500, 'total_timesteps': 10, 'main_seed': 12345}
Logging to ../exp/tutorial_model_no-cheating-deterrent
Training took 0.0022253990173339844 seconds.
This was not a full training as we only trained for 10 steps!
```

```
Increase config.total_timesteps for an actual training.
You can load this model by using the name 'tutorial_model_no-cheating-deterrent'
```

Now you should have seen everything you need to start diving into simulating crowdworker behavior. You can modify the settings and see how this affects the behavior of the crowdworker. You can find further information on the specific classes in the code's documentation.

If you have any questions or issues, feel free to reach out to us!