**IJCAI 2025 Submission #5415 Code and Data Instruction**

This is the code and data instruction for IJCAI 2025 submission #5415 titled

*Counterfactual Explanations for Continuous Action Reinforcement Learning*

This instruction is for reproducing all the experiment results step by step. The code and results of each step can be found in the corresponding subfolder. There is no need to change the hyper parameters in the code if you want to reproduce the results, as those are used in the experiments part of the paper.

* To generate results reported in Section 5.1 of the paper, please follow steps in Diabetes Case Study.
* To generate results reported in Section 5.2 of the paper, please follow steps in Lunar Lander Case Study.
* To generate analysis results and figures, please follow steps in data postprocess

**Diabetes Case Study**

1. **Train baseline PPO**

We train a PPO model using Stable-Baselines3 to generate original traces and serve as the baseline.

1. Build conda environment, install necessary packages and replace with given file

conda create -n CF\_diabetic\_train\_ppo python=3.8

conda activate CF\_diabetic\_train\_ppo

pip install stable-baselines3==1.7.0

pip install gym==0.21.0

pip install torch==2.0.0

pip install pandas==1.5.3

pip install tensorboard

1. Update packages with given file

cd yourpath/.conda/envs/ CF\_diabetic\_train\_ppo/lib/python3.8/site-packages

Upload packages in train\_ppo/package to replace default ones.

1. For single-environment simulation, adult patient # 7 was selected, and the PPO model was trained for 100,000 steps with a learning rate of 0.0001 and a gradient step size of 50 with the following command, the other parameters are set in code.

For multi-environment simulation, adult patients #4, 5, 6 were selected for training, and adult patients #7, 8, 9 for testing. Each patient environment underwent 3,000 training steps per round, followed by 3,000 testing steps per environment. A total of 179 training rounds were completed. Before running code, please replace the file path to your local path. To train baseline PPO, use the following command:

**Single-environment:**

python diabetic\_case\_train\_ppo.py -arg\_patient\_type adult -arg\_patient\_id 7 -arg\_cuda 0 -arg\_train\_step 100000 -arg\_callback\_step 100000

**Multi-environment:**

python diabetic\_case\_train\_ppo\_generalize.py -arg\_exp\_id 1 -arg\_cuda 0 -arg\_train\_step\_each\_env 3000 -arg\_callback\_step 3000 -arg\_train\_round 179 -arg\_lr 0.0001 -arg\_test\_epochs\_each\_env 1 -arg\_max\_test\_time\_each\_env 3000

1. The trained models are in folder: train\_ppo/trained\_model
2. **Run experiment with given original trace**
3. Build conda environment, install necessary packages and replace with given file

conda create -n CF\_diabetic python=3.8

conda activate CF\_diabetic

pip install stable-baselines3==1.7.0

pip install gym==0.21.0

pip install torch==2.0.0

pip install pandas==1.5.3

1. Update packages with given file

cd yourpath/.conda/envs/CF\_diabetic /lib/python3.8/site-packages

Upload packages in train\_td3/package to replace default ones.

1. Run code to produce experiment results

For single-environment simulation, adult patient # 7 was selected. For multi-environment, adult # 4,5,6 are selected for training and adult # 7,8,9 for testing.

In the diabetic patient simulation, the code excluded segments meeting one of the following criteria: (1) total reward equals 20 (the maximum possible reward, making total reward improvement unnecessary), (2) initial CGM value is below 65 (indicating meal intervention, which is not modeled as an action), or (3) all actions are zero (results driven by randomness). The remaining segments were grouped based on their initial state values into intervals of (65, 100], (100, 150], and (150, 260]. From each group, 6 segments were selected for training and 6 for testing, resulting in 18 segments each for the training and testing sets.

We use SLURM to run experiments in batch. The parameters used are in folder

train\_td3/code/P1, train\_td3/code/P2-base, train\_td3/code/P2-fixed for each research problem. For those not included in the txt, please see diabetic\_case\_cf\_server\_fix\_all\_trace\_sb3\_fix\_action\_step\_user\_input.py for default values.

Please use the following command to run the experiment with existing original traces. The example original trace file data are in folder train\_td3/original\_trace (only part of the data is included due to file size limit). Before running code, please replace all file path in the code to your corresponding local path.

**For P1:**

P1 single environment: sbatch run\_diabetic\_exp1.sh

P1 multi environment: sbatch run\_diabetic\_exp2.sh

**For P2-base:**

P2-base single environment: sbatch run\_diabetic\_exp3\_State\_RP2.sh with diabetic\_experiments\_exp1\_state\_RP2\_UET11.txt

P2-base multi environment: sbatch run\_diabetic\_exp3\_State\_RP2.sh with diabetic\_experiments\_exp3\_state\_RP2\_UET6.txt

**For P2-fixed:**

P2-fixed single environment: sbatch run\_diabetic\_exp3\_State\_RP3.sh with diabetic\_experiments\_exp1\_state\_RP3\_UET11.txt

P2-fixed multi environment: sbatch run\_diabetic\_exp3\_State\_RP3.sh with diabetic\_experiments\_exp3\_state\_RP3\_UET6.txt

1. **Run experiment with new original trace**

If you want to test the code with new traces, please use script\_diabetic\_exp\_multiple\_trace.py to produce the parameter txt files with your chosen parameters, apply them in the abovementioned .sh files to run new experiments. Examples are in train\_td3\code\P1\generate\_new\_trace

1. **Result data example**

Due to the limit of supplementary file, we only include part of the generated traces and results here. They can be found in folder train\_td3\sample\_data

1. **Data postprocess**
2. Generate result in Table 1 and Table 2

For experiment results of 1 group of original trace, 7 trials are collected. First, rename the data folders trial\_1 in folder ‘td3\_cf\_results’ trial\_X (X:1-7), then move folders trial\_2~trial\_7 to folder ‘td3\_cf\_results’ of trial\_1 to put them all together.

After that, open file data\_postprocess.py, assign case\_name = 'diabetic' on line 3059. To get the preliminary results, run code block under comment ‘basic postprocess to get all metric data’. Please assign the corresponding parameters of the experiment as in the above txt files and your own file path.

The result files generated are:

all\_metric\_aveg\_best\_baseline\_test.csv, all\_metric\_aveg\_with\_pointwise\_test.csv

Add suffix to those two files to include the experiment parameter. For example:

all\_metric\_aveg\_best\_baseline\_test\_0\_RP1.csv: for P1, without any extra constraint

all\_metric\_aveg\_best\_baseline\_test\_100\_RP2.csv: for P2-base, set state threshold 100

all\_metric\_aveg\_best\_baseline\_test\_100-0.03\_RP3.csv: for P2-fixed, set state threshold 100 and fixed action value 0.03

Data in the tables are in all\_metric\_aveg\_best\_baseline\_test.csv, in which ‘P\_hr’ value is for metric rho\_plus and ‘compare\_count\_perc’ is for metric rho\_adv in the paper.

1. Generate Figure 2 and Figure 3

Build a new folder ‘across\_rp’. Move all generated result files of the three research problems (P1, P2-base, P2-fixed) to it. Run code block under comment ‘draw final learning curve’ after assign case name and your local file path. Assign metric\_name=’rho\_plus’ for Figure 2, and metric\_name=’rho\_adv’ for Figure 3

1. The final result data and figures used in the paper are in folder: Supplementary\code\_and\_data\data\_postprocess\Diabetes\_case\_study\across\_rp\final\_data

**Lunar Lander Case Study**

For lunar lander case study, the steps are similar to the diabetes case study.

1. **Train baseline PPO**

We trained a PPO model using Stable-Baselines3 to generate original traces and serve as the baseline.

1. Build conda environment, install necessary packages and replace with given file

conda create -n CF\_LunarLander\_train\_ppo\_generalize python=3.8

conda activate CF\_LunarLander\_train\_ppo\_generalize

pip install stable-baselines3==1.7.0

pip install gym==0.21.0

pip install torch==2.0.0

pip install pandas==1.5.3

pip install tensorboard

1. Update packages with given file

cd yourpath/.conda/envs/CF\_LunarLander\_train\_ppo\_generalize/lib/python3.8/site-packages

Upload packages in train\_ppo/package to replace default ones.

1. For single-environment simulation, we used one environment with g=-9, training the PPO model for 3000 steps with a learning rate of 0.0001 and a gradient step size of 20. For multi-environment, three training environments (g=-11, -9, -5) were used, along with three distinct test environments (g=-10, -8, -6). The PPO model was trained in each environment for 500 steps per round and tested for 1,800 steps per environment after each round. The training spanned 3 rounds. The other parameters are set in code. Before running code, please replace the file path to your local path.

**Single-environment:**

python openai\_case\_train\_ppo\_generalize.py -arg\_exp\_id 1 -arg\_cuda 0 -arg\_train\_step\_each\_env 500 -arg\_callback\_step 500 -arg\_train\_round 5 -arg\_lr 0.0001 -arg\_test\_epochs\_each\_env 1 -arg\_max\_test\_time\_each\_env 1800 -arg\_if\_train\_personalize 1 -arg\_assigned\_gravity -9.0

**Multi-environment:**

python openai\_case\_train\_ppo\_generalize.py -arg\_exp\_id 1 -arg\_cuda 0 -arg\_train\_step\_each\_env 500 -arg\_callback\_step 500 -arg\_train\_round 3 -arg\_lr 0.0001 -arg\_test\_epochs\_each\_env 1 -arg\_max\_test\_time\_each\_env 1800

1. The trained models are in folder: train\_ppo/trained\_model
2. **Run experiment with given original trace**
3. Build conda environment, install necessary packages and replace with given file

conda create -n CF\_lunarlander python=3.8

conda activate CF\_lunarlander

pip install stable-baselines3==1.7.0

conda install swig

pip install box2d-py

pip install gym==0.21.0

pip install Box2D

pip install torch==2.0.0

pip install pandas==1.5.3

pip install tensorflow==2.13.0

1. Update packages with given file

cd yourpath/.conda/envs/CF\_lunarlander/lib/python3.8/site-packages

Upload packages in train\_td3/package to replace default ones.

1. Run code to produce experiment results

For single-environment simulation, g=-9 was selected. For multi-environment, three training environments (g=-11, -9, -5) were used, along with three distinct test environments (g=-10, -8, -6).

We use SLURM to run experiments in batch. The parameters used are in folder

train\_td3/code/P1, train\_td3/code/P2-base, train\_td3/code/P2-fixed for each research problem. For those not included in the txt, please see openai\_case\_cf\_server\_fix\_all\_trace\_sb3\_fix\_action\_step\_user\_input.py for default values.

Please use the following command to run the experiment. To reproduce the results in the paper, please use the provided original trace files. The example original trace file data are in folder train\_td3/original\_trace. Before running code, please replace all file path in the code to your corresponding local path.

**For P1:**

P1 single environment: sbatch run\_LL\_exp1\_double\_test.sh

P1 multi environment: sbatch run\_LL\_exp3\_double\_test.sh

**For P2-base:**

P2-base single environment: sbatch run\_LL\_RP2\_exp1\_S\_double\_test.sh

P2-base multi environment: sbatch run\_LL\_RP2\_exp3\_S\_double\_test.sh

**For P2-fixed:**

P2-fixed single environment: sbatch run\_LL\_RP3\_exp1\_S\_double\_test.sh

P2-fixed multi environment: sbatch run\_LL\_RP3\_exp3\_S\_double\_test.sh

1. **Run experiment with new original trace**

If you want to test the code with new traces, please use script\_LL\_exp.py to produce parameters in txt files with your chosen parameters, apply them in the abovementioned .sh files to run new experiments. Examples are in train\_td3\code\P1\generate\_new\_trace

1. **Result data example**

Due to the limit of supplementary file, we only include part of the generated traces and results here. They can be found in folder train\_td3\sample\_data

1. **Data postprocess**
2. Generate result in Table 1 and Table 2

For experiment results of 1 group of original trace, 7 trials are collected. First, rename the data folders trial\_1 in folder ‘td3\_cf\_results’ trial\_X (X:1-7), then move folders trial\_2~trial\_7 to folder ‘td3\_cf\_results’ of trial\_1 to put them all together.

After that, open file data\_postprocess.py, assign case\_name = ' lunar\_lander' on line 3059. To get the preliminary results, run code block under comment ‘basic postprocess to get all metric data’. Please assign the corresponding parameters of the experiment as in the above txt files and your own file path.

The result files generated are:

all\_metric\_aveg\_best\_baseline\_test.csv, all\_metric\_aveg\_with\_pointwise\_test.csv

Add suffix to those two files to include the experiment parameter. For example:

all\_metric\_aveg\_best\_baseline\_test\_0.0\_RP1: for P1, without any extra constraint

all\_metric\_aveg\_best\_baseline\_test\_0.18\_RP2.csv: for P2-base, set state threshold [-0.18, 0.18]

all\_metric\_aveg\_best\_baseline\_test\_0.18-0\_0\_RP3.csv: for P2-fixed, set state threshold [-0.18, 0.18] and fixed action value [0,0]

Data in the tables are in all\_metric\_aveg\_best\_baseline\_test.csv, in which ‘P\_hr’ value is for metric rho\_plus and ‘compare\_count\_perc’ is for metric rho\_adv in the paper.

1. Generate Figure 4 and Figure 5

Build a new folder ‘across\_rp’. Move all generated result files of the three research problems (P1, P2-base, P2-fixed) to it. Run code block under comment ‘draw final learning curve’ after assign case name and your local file path. Assign metric\_name=’rho\_plus’ for Figure 4, and metric\_name=’rho\_adv’ for Figure 5

1. The final result data and figures used in the paper are in folder: Supplementary\code\_and\_data\data\_postprocess\lunar\_lander\across\_rp\final\_data