Documentation

Create environment

Install conda:

- download and install Anaconda3 for Linux .sh

Open terminal create and activate new environment:

```
$ conda create --name myenv
$ conda activate myenv
```

Install the next packages with

```
$ pip3 install tensorflow
$ pip3 install keras
```

Don't have cmake installed? No problem:

Install build tools and libraries that CMake depends on:

```
$ sudo apt-get install build-essential libssl-dev
```

Go to the temp directory:

```
$ cd /tmp
```

Then, enter the following command to download the source code:

```
$ wget https://github.com/Kitware/CMake/releases/download/v3.20.0/cmake-3.20.0.tar.gz
```

Once the tar.gz file is downloaded, enter the following command to extract it:

```
$ tar -zxvf cmake-3.20.0.tar.gz
```

Then move to the extracted folder as follows:

```
$ cd cmake-3.20.0
```

Finally, run the following commands to compile and install CMake:

```
./bootstrap
```

The bootstrap process may take some time, do not interrupt it. Then you can make it using the following command:

```
$ make
```

And then install it as follows:

```
$ sudo make install
```

How to launch the program

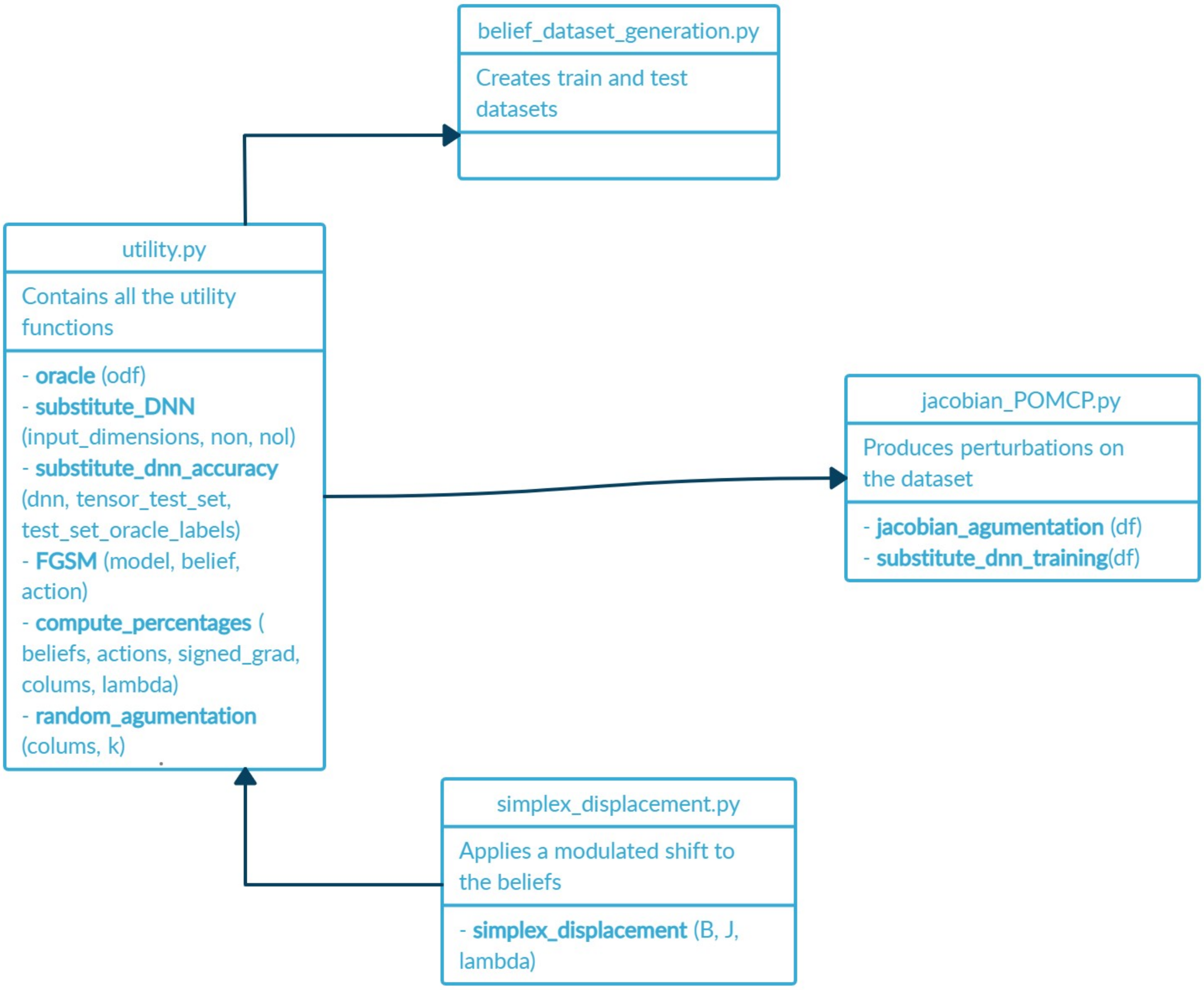
Open terminal and create new dir and enter it

```
$ mkdir mydir
$ cd mydir
```

Then:

- clone github repository -> https://github.com/GiuMaz/pomcp-blackbox
- enter cloned repository
- create new dir "mkdir build" and enter it "cd build"
- cmake -DCMAKE_BUILD_TYPE=Release ..
- make

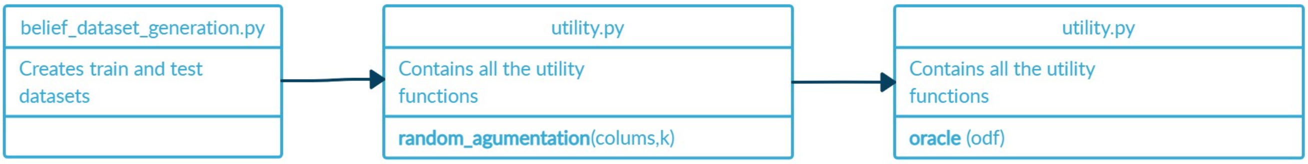
Repository's composition


belief_dataset_generation.py
Creates train and test datasets
utility.py
Contains all the utility functions
- oracle (odf)
- substitute_DNN (input_dimensions, non, nol)
- substitute_dnn_accuracy (dnn, tensor_test_set, test_set_oracle_labels)
- FGSM (model, belief, action)
- compute_percentages (beliefs, actions, signed_grad, colums, lambda)
- random_agumentation (colums, k)
jacobian_POMCP.py
Produces perturbations on the dataset
- jacobian_agumentation (df)
- substitute_dnn_training(df)
simplex_displacement.py
Applies a modulated shift to the beliefs
- simplex_displacement (B, J, lambda)

!!! Remember to change the "path = .." variable where you find it !!!

belief_dataset_generation.py

This is used to generate the train_set (80 beliefs) and the test_set (300 beliefs).
Here random_agumentation is used to create the dataframes of train and test.

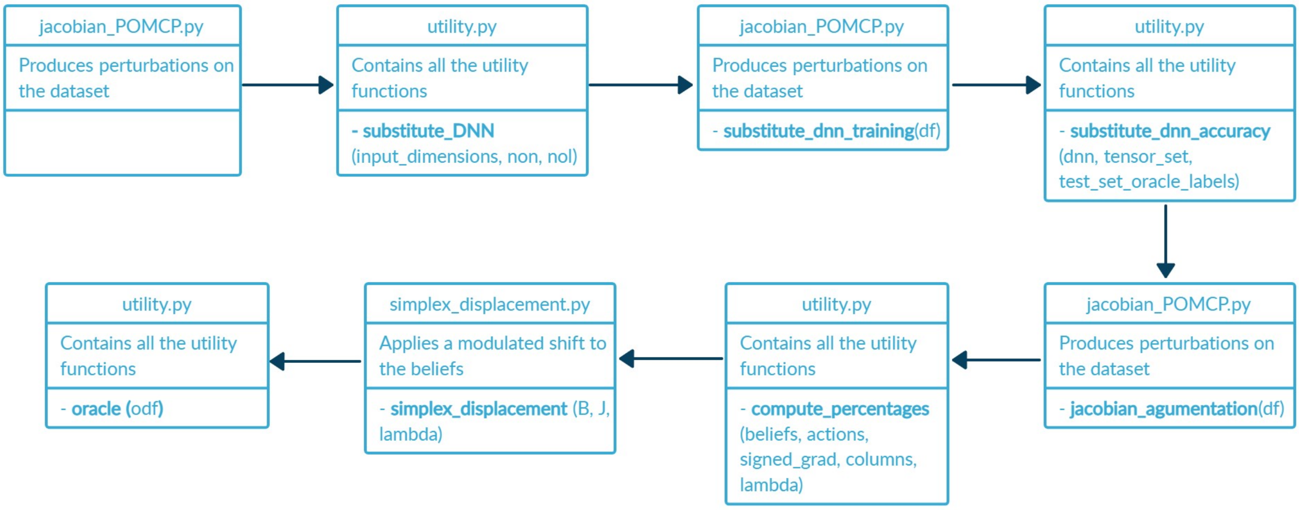

belief_dataset_generation.py
Creates train and test datasets
utility.py
Contains all the utility functions
random_agumentation(colums,k)
utility.py
Contains all the utility functions
oracle (odf)

jacobian_POMCP.py

In this file the jacobian based black-box attack on the POMCP model is implemented. For the POMCP integration a python wrapper is used which passes a belief dataset to the POMCP Oracle (C++ exe) to obtain the relative actions (labels) in output.
The outputs are:

- train dataset is agumented
- dnn is trained

- list of accuracies for each iteration


jacobian_POMCP.py
Produces perturbations on the dataset
utility.py
Contains all the utility functions
- substitute_DNN (input_dimensions, non, nol)
jacobian_POMCP.py
Produces perturbations on the dataset
- substitute_dnn_training(df)
utility.py
Contains all the utility functions
- substitute_dnn_accuracy (dnn, tensor_set, test_set_oracle_labels)
jacobian_POMCP.py
Produces perturbations on the dataset
- jacobian_agumentation(df)
utility.py
Contains all the utility functions
- compute_percentages (beliefs, actions, signed_grad, columns, lambda)
simplex_displacement.py
Applies a modulated shift to the beliefs
- simplex_displacement (B, J, lambda)
utility.py
Contains all the utility functions
- oracle (odf)

simplex_displacement.py

Applies a modulated shift to the belief so as to remain within the simplex without breaking the constraints of the probability distribution.

utility.py

Library containing all the utility functions necessary for the project.

1. **oracle**: a python wrapper for the C++ executable representing the oracle (POMCP)
2. **substitute_DNN**: creates a deep neural network with 'nol' layers and 'non' neurons for each layer
3. **substitute_dnn_accuracy**: calculates the accuracy of the substitute network with respect to the labels classified by the oracle
4. **FGSM**: generates the FGSM attack by exploiting the internal parameters of the substitute network to obtain an adversarial example on a belief that will then be used to deceive the POMCP thanks to the transferability properties of the attacks.
5. **compute_percentages**: moves every belief towards the decision boundaries of the POMCP according to the simplex_displacement protocol in order to obtain a change of action, furthermore the percentages for each label are calculated
6. **random_agumentation**: does not apply the displacement via Jacobian but adds at each iteration a set of random beliefs generated by dirichlet distribution to obtain a basic result on which to compare the performance of other displacement methods.