Getting started with ACE

Requirements: Docker, Docker-compose

Suggestion for testing and configuring experiment: Postman or curl

OBS: The frontend was not working properly so I disabled it at the moment. So I will only cover using the APIs directly.

## Clone the repository

from ‘’

## Launch the image

* without building (in the correct folder): ‘docker-compose up’
* Force building (in the correct folder): ‘docker-compose up --build’
* Building without cache: ‘docker-compose build –no-cache’

## Test connection

* + HTTP GET request to ‘http://127.0.0.1:5000/api/ping/<AnyWord>’

curl -X GET \

http://127.0.0.1:5000/api/ping/Testing \

-H 'cache-control: no-cache'

* + Successful response: ‘I am here! Ping: Testing’

## Create an experiment

* + If you create an experiment if the same name as other, we will reset the statistical model for the future interactions. But all the collected data and the statistical models are kept saved in the database. But I still haven’t implemented a way to restore the model…
  + HTTP POST request to ‘http://127.0.0.1:5000/api/blackbox/configure\_job’
  + POST payload is the job configuration in JSON
  + If it was successful: “Ok”

{

"job":"testlghoo-optace",

"unit\_diversion": "id",

"signals":["reward","x1","x2"],

"algorithm": {

"type":"LGHOO",

"objective": ["reward"],

"v1":1.0,

"rho":0.5,

"context":[],

"minimum\_grow":1,

"dimensions":[{

"name":"x1",

"height\_limit":10,

"arm\_min":0,

"arm\_max":5

},

{

"name":"x2",

"height\_limit":10,

"arm\_min":0,

"arm\_max":2

}]

}

}

Some explanation:

* **job**: (String) the name of the optimization procedure. This should be a unique name that will differentiate this optimization from the others
* **unit\_diversion**: (String) This is the type of identifier that will be used in the randomization of some types of experiments. E.g. if you want user consistency in an A/B test it is good to specify what is going to be the parameter here. This is not used for all types of experiment.
* **signals**: (Array of strings) all variables that are going to be logged in the database and possibly used in the experiment. The reward/objective variable should be here (this is the metric that you are trying to maximize). Other signals can correspond to variables that could be interesting to analyze later, and the name of the variables you are trying to optimize.
* **algorithm**: (JSON dictionary) Specifies and configures the optimization algorithm
  + **type**: (String) the name of the algorithm. There are a few algorithms implemented: “lghoo”, “doo”, “ucb1”. There are others but those where not well tested yet (“epsilon-greedy”, “epsilon-greedy-annealing”, “softmax”, “softmax-annealing”, “ucb2”)
  + **objective**: (Array of strings) Here you specify the names of the different objectives you are tracking. At the moment, it only works with a single objective (the first in the array).
  + **context**: (Array of strings) Depending on the algorithm you might want to add contextual information (you add them in this array)
  + **v1, rho** and **minimum\_grow**: (Float) Those are parameters specific for the LGHOO algorithm (there are sensible default values). Different algorithms will require different parameters.
  + **dimensions**: (Array of JSON dictionaries) Here we specify characteristics of the dimensions we are optimizing. E.g. for optimization in a range in the LGHOO we specify the range (max and min arm), the precision (height limit). If we are running a regular multi-armed bandit, it has a single dimension, and the others will be ignored
    - **name**: (String) this is the required value for every dimension.

## Update the model/Log information

* + HTTP POST request to ‘http://127.0.0.1:5000/api/blackbox/update\_model’
  + POST payload is the job configuration in JSON
  + Success: “Ok”

{

"job": "testlghoo-optace",

"unit\_diversion": "id-1",

"signals" : {

"reward": 0.5,

"x1": 2.5,

"x2": 1.0

}

}

## Request new trials/new variations

* HTTP POST request to ‘http://127.0.0.1:5000/api/blackbox/request\_trial’
* POST payload is the job configuration in JSON

{

"job": "testlghoo-optace",

"unit\_diversion": "id-1"

}

* If successful we get a reply in JSON with the new trials per dimension. E.g.:

{"x1": 2.5, "x2": 1.0}

## Check which was the best value (at the moment of request).

* Note that here you might not want to ask this at any point. If you start the experiment and run this after only a few interactions (e.g. 100 data points) you might get incorrect or biased results. The ideal is to first determine a minimum number of data points that you want and only request after that
* HTTP POST request to ‘http://127.0.0.1:5000/api/blackbox/get\_best\_arm’
* POST payload is the job configuration in JSON

{

"job": "testlghoo-optace"

}

* If successful:

{"x1": 2.5, "x2": 1.0}