Bender Optimizer Documentation

Release latest

Jul 03, 2019

Introduction

CHAPTER 1

Bender Clients Status

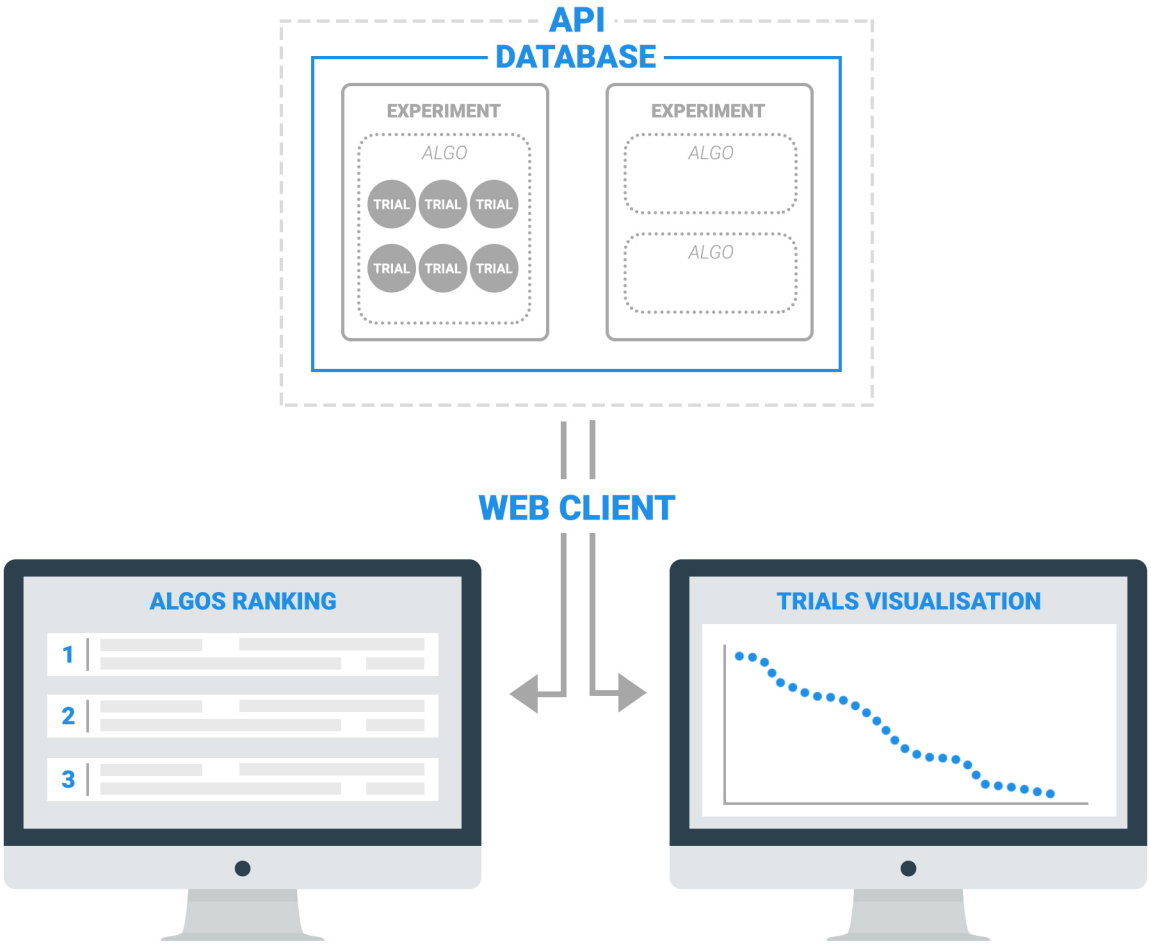
CHAPTER 2

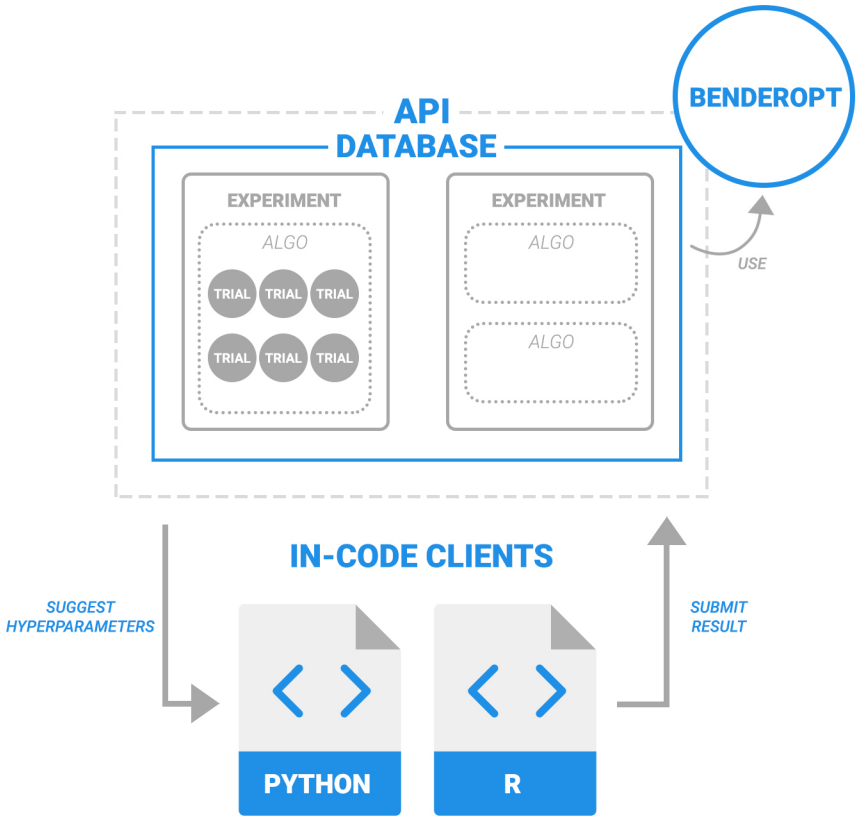
About Bender

Bender is a hyperparameters optimizer for black box optimization. It is currently open-access but we plan to open-source all its components.

It features 3 kind of components:

- ***benderopt*: A custom library implementing 2 algorithms:**
  - Random Search
  - Parzen Estimator Optimization (TPE-like)
- ***bender-api*: A high level interface to interact with bender functionalities.**
  - CRUD for experiments, algorithms and trials.
  - Relies on benderopt for suggesting new hyperparameters to try.
- ***Various clients* to give you access to Bender functionalities from anywhere:**
  - *web-client* available at `https://bender.dreem.com`
  - *python-client* available by `pip install bender-client`


API
DATABASE
EXPERIMENT
ALGO
TRIAL
TRIAL
TRIAL
TRIAL
TRIAL
TRIAL
EXPERIMENT
ALGO
ALGO
WEB CLIENT
ALGOS RANKING
1
2
3
TRIALS VISUALISATION



You can find a medium post about bender here: https://medium.com/@dylanheirstraeten/bender-c394a5bda78b

CHAPTER 3

Use bender locally

If you don't want to use our online services, you still have the option to run bender locally using docker.

Prerequisites: docker and git

To do so, first clone, fork, or download this repository, and from its root run the following command lines :

(**WARNING**: *This will be done with default settings, to change the installation and initialisation settings please scroll to the next header*)

`make install` : To download all the ressources needed and install all bender dependencies into ./local

`make build` : To build the different dependencies.

`make start` : To run the database, the API and the frontend.

`make` : Tu do the 3 above commands at the same time.

`make stop` : To stop all bender services.

`make clear` : To remove all bender services, source files, images and containers.

To use the different clients with the local API check the its specific documentation.

CHAPTER 4

Configure bender locally

Bender will use the configuration file in `.benderconf`.

It is defined in JSON syntax and looks like the following :

```
{
    "benderWebClient": {
        "localhostPort": <the port you want to use on localhost to expose the web-
↪client>
    },
    "benderApi": {
        "localhostPort": <the port you want to use on localhost to expose the api>
    },
    "benderDatabase": {
        "username": <the username of the admin acount to create in the database>,
        "email": <the email of the admin acount to create in the database>,
        "password": <the password of the admin acount to create in the database>,
        "databasePath": <the local volume to mount on the database docker to host␣
↪your database>,
        "localhostPort": <the port you want to use on localhost to expose the␣
↪database>
    }
}
```

CHAPTER 5

Table Of Content

5.1 Concepts

With Bender, you create **Algos** within **Experiments**, and ask Bender some **Suggestions** of Hyperparameters to try for these **Algos**, benchmark your algo with these hyperparameters, and then submit the results as a **Trial**.

You can then compare the **Trials** to get the best set of Hyper-Parameters for a given **Algo** or compare your **Algos** and get the best results possible for your **Experiment**!

5.1.1 Experiment

An **Experiment** is closely tied to a problem.

You want for example to recognize automatically handwritten digits? That is a problem: you can create an **Experiment** for example named *digits_recogition*.

You want to find the shortest path to go somewhere? That's another problem: therefore you create another **Experiment**.

Every experiment has an unique id in the form of: `0597ca48-66f7-42be-9021-12ec57d4251e`

Every bender client allows you to create, load or delete an **Experiment**, just check out the specific documentation to learn how.

5.1.2 Algo

An Algo is simply a way to respond to an experiment: a way to answer the given problem.

Getting back to our example of digit recognition problem, there is plenty different ways to solve the problem: one would be a certain kind of neural network, another would be a random forest algorithm, etc.

Some are better than others but Bender is here to maximise the performances of each **Algo** and allows you to also compare them to find the best way to answer your **Experiment**.

For each **Algo** you want to specify a set of Hyperparameters that Bender will optimize. These parameters can be the learning rate of your neural network, the number of trees in your random forest, etc.

5.1.3 Trial

A **Trial** is Bender's food. To train Bender on an Algo you created, he needs data to improve himself.

Each time you try a set of Hyperparameters, you want to make Bender know about it, also giving him a performance indicator associated with this Hyperparameters set (a loss, an accuracy, etc.).

In short: a **Trial** is a Hyperparameters set associated with a performance metric.

5.1.4 Suggestion

Of course it's not up to you to decide which Hyperparameters to use or not. Just ask Bender to give you some new ones: that is a **Suggestion**.

Therefore you can automate the whole process of optimizing your hyperparameters efficiently and quicky.

5.2 Use the web client

The web client is here to make you life easier. Use it.

It's the only client that you can use to compare all your **Trials** and all your **Algos** by visualizing them on graphs and ranking boards.

Let's make a tour of the web client :

5.2.1 Login

First, on bender.dreem.com, log yourself in or create an account if you don't already have one.

You will then be redirected to the **Dashboard**.

5.2.2 Dashboard

The dashboard is the heart of the web client and is divided in different parts:

5.2.3 The menu

Here you can perform some basic actions like looking at your profile informations and update them, loging out or contact us.

You can hide or show this menu by clicking on the little arrow on the side.

5.2.4 The dynamic breadcrumb

You can find here all the usefull informations about the **Board** you are curently on.

Starting with a welcome message, it also can display informations about the current **Experiment** or **Algo**.

You can also navigate between the **Boards** by cliking the desired element.

At the far right you will find either the ID of the **Experiment** or **Algo** that is displayed on the **Board** : just click it to copy an use it in the other clients.

5.2.5 The boards

Again, within the boards we can identify different elements :

5.2.6 The experiments board

On this **Board** you can see a simple list of the **Experiments** that you currently own with some basic infomations related.

Either click on one to switch to the **Ranking Board** of this **Experiment** or just delete it by cliking on the red button that appears on the right when you hover it.

To create a new **Experiment** just click the green plus sign at the bottom of the list.

5.2.7 The ranking board

Here you can see all your different **Algos** ranked by a selected performance metric.

By clicking a column title you can rearange the ranking depending on this new metric.

By clicking a cell you can display the specific set of hyperparameters used to get this result.

To create a new **Algo** just click the green plus sign at the bottom.

On **Algo** cell hover, two buttons appears : edit and delete.

And still within the **Algo** cell you have 2 buttons to enter the **Trials Board** and explore all your **Trials** either with a line chart or a scatter chart.

5.2.8 The trials board

The final board : two modes depending on the botton you clicked on the **Ranking Board**, line or scatter views.

In both case you can click points on the graph to display the **Trial** associated results and used hyperparameters.

5.3 API

The easiest way to use and test our API is with the following **Postman** collection where you should find everything you need.

Just a friendly warning to tell you that postman currently doesn't handle automatic cookies clearing and that if you were in any way having some troubles with our collection, just try to wipe cookies manually with postman's UI.

5.3.1 How to use it

Install postman by clicking here.

Import our collection following these steps.

Get the postman json file you need to import over here.

5.4 Python

The **python client for bender** is available with the `pip install bender-client` command.

You can find a full tutorial on the github repo of the package at: github.com/Dreem-Organization/Bender-Client.

Here is a minimal example:

```
""" In this minimal example we try to minimize the sinus function between 0 and 2pi"""

from benderclient import Bender
import numpy as np

# Initialize Bender
bender = Bender()

# Create an experiment
bender.create_experiment(
    name='Minimum example experiment',
    description='Find minimum of sinus function',
    metrics=[{"metric_name": "sinus_value", "type": "loss"}],
)

# Create an algo (here the sinus function with one parameter)
bender.create_algo(
    name='Analytic sinus function',
    hyperparameters=[
        {
            "name": 'x',
            "category": "uniform",
            "search_space": {
                "low": 0,
                "high": 2 * np.pi,
            },
        },
    ]
)

# Ask bender for values to try
for _ in range(50):
    # Get a set of Hyperparameters to test
    suggestion = bender.suggest(metric="sinus_value", optimizer="parzen_estimator")

    # Run the sinus function
    sinus_value = np.sin(suggestion["x"])

    # Feed Bender a trial
    bender.create_trial(
        hyperparameters=suggestion,
```

(continues on next page)

(continued from previous page)

```
        results={"sinus_value": sinus_value}
    )
    print("x: ", suggestion["x"], " value :", sinus_value)
```

5.5 API

PLEASE READ THE 'CONCEPTS' DOCUMENTATION TAB FIRST IF IT'S YOUR FIRST TIME USING THE API

5.5.1 Base route

All routes starts with the following base : `https://bender-api.dreem.com` (of your own API address if you are running Bender yourself)

The Api is using JWT authentication so don't forget to sent the token in your request headers (You can get it with the login route).

Make sure you already created a Bender account on the web client !

5.5.2 Login

POST `/login`

Body

```
{
    "email": "my@mail.com",
    "password": "mydeepestsecret",
}
```

Return

```
{
    "token": "FNU87TR875R7IFO873F8E873T",
    "user": {
        "pk": 83,
        "username": "my_user",
        "email": "my@mail.com",
    }
}
```

5.5.3 Experiment methods

List experiments

GET `/api/experiments/?owner=<username_of_your_account>`

Return

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": "0555ce2c-25b9-4804-b7a6-6fb6fb069e3a",
            "name": "MY_EXPERIMENT",
            "description": "none",
            "metrics": [
                {
                    "type": "loss",
                    "metric_name": "A"
                },
                {
                    "type": "reward",
                    "metric_name": "B"
                }
            ],
            "dataset": "MY_DATASET",
            "dataset_parameters": null,
            "trial_count": 6,
            "algo_count": 2,
            "owner": "MY_USERNAME",
            "participants": [
                [
                    "MY_USERNAME"
                ]
            ],
            "shared_with": [],
            "created": "2019-01-14T14:26:28.289729Z",
            "modified": "2019-01-14T14:26:28.289786Z"
        }
    ]
}
```


Create experiment

POST `/api/experiments/`

Body

```
{
    "name": "my_experiment_name",
    "description": "my_experiment_description",
    "metrics": [{"metric_name": "metric_a","type": "reward" }, {"metric_name":
↪"metric_b","type": "loss" }],
    "dataset": "my_dataset_name",
    "dataset_parameters": { "version": 0.1, "anything": "you_want" }
}
```

Return


```
{
    "id": "fb2c7cb6-c1d9-4b4f-8547-9064485673aa",
    "name": "my_experiment_name",
```


(continues on next page)

(continued from previous page)

```
    "description": "my_experiment_description",
    "dataset": "my_dataset_name",
    "dataset_parameters": {
        "version": 0.1,
        "anything": "you_want"
    },
    "metrics": [
        {
            "type": "reward",
            "metric_name": "metric_a"
        },
        {
            "type": "loss",
            "metric_name": "metric_b"
        }
    ]
}
```

Delete experiment

DELETE `/api/experiments/<your_experiment_id>/`

Return :

none

5.5.4 Algo methods

List algos

GET `/api/algos/?experiment=<experiment_id>`

Return

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": "1221bd9c-634a-4257-9419-dce9fd7581ec",
            "name": "my_algo_name",
            "experiment": "e0a1c447-f7d9-43e3-8b8e-53aa2806db8d",
            "owner": "ibob",
            "parameters": [
                {
                    "algo": "1221bd9c-634a-4257-9419-dce9fd7581ec",
                    "name": "param_name",
                    "category": "categorical",
                    "search_space": {
                        "values": [
                            3,
                            5,
```

(continues on next page)

(continued from previous page)


```
                            7
                        ]
                    },
                    "category_display": "Categorical parameter"
                }
            ],
            "description": "my_algo_description",
            "trial_count": 0,
            "created": "2019-01-21T14:40:14.429439Z",
            "modified": "2019-01-21T14:40:14.429471Z",
            "is_search_space_defined": true
        }
    ]
}
```


Create algo

POST `/api/algos/`

Body


```
{
    "name": "my_algo_name",
    "description": "my_algo_description",
    "parameters": [{"name": "param_name","category": "categorical", "search_space": {
↪"values": [3, 5, 7] } }]
    "experiment": "your_experiment_id"
}
```


Return


```
{
    "id": "1221bd9c-634a-4257-9419-dce9fd7581ec",
    "name": "my_algo_name",
    "experiment": "e0a1c447-f7d9-43e3-8b8e-53aa2806db8d",
    "description": "my_algo_description",
    "parameters": [
        {
            "name": "param_name",
            "category": "categorical",
            "search_space": {
                "values": [
                    3,
                    5,
                    7
                ]
            }
        }
    ],
    "is_search_space_defined": true
}
```


Get suggestion from bender

POST `/api/algos/<my_algo_id>/suggest`

Body

```
{
    "metric": "metric_a",
    "optimizer": "parzen_estimator"
}
```

Return

```
{
    "param_name": 7
}
```

Delete algo

DELETE `/api/experiments/<your_algo_id>/`

Return :

none

5.5.5 Trials methods

List trials

GET `/api/trials/?algo=<algo_id>`

Return

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": "adf5bf14-0fb1-4e96-8e37-ddd4e5c8f44b",
            "algo": "1221bd9c-634a-4257-9419-dce9fd7581ec",
            "experiment": "e0a1c447-f7d9-43e3-8b8e-53aa2806db8d",
            "owner": "ibob",
            "parameters": {
                "param_name": 3
            },
            "results": {
                "metric_a": 0.8,
                "metric_b": 0.3
            },
            "comment": "anything_you_want_to_say",
            "algo_name": "my_algo_name",
            "created": "2019-01-21T14:52:40.069199Z",
            "modified": "2019-01-21T14:52:40.069230Z",
            "weight": 1
        }
    ]
}
```

Create trial

POST `/api/trials/`

Body

```
{
    "algo": "my_algo_id",
    "description": "my_algo_description",
    "parameters": {"param1": "value","param2": 3567, "param3": "another"},
    "results": {"metric1": 0.8, "metric2": 0.3},
    "comment": "anything_you_want_to_say",
    "weight": 1
}
```

Return

```
{
    "id": "adf5bf14-0fb1-4e96-8e37-ddd4e5c8f44b",
    "algo": "1221bd9c-634a-4257-9419-dce9fd7581ec",
    "parameters": {
        "param_name": 3
    },
    "results": {
        "metric_a": 0.8,
        "metric_b": 0.3
    },
    "comment": "anything_you_want_to_say",
    "weight": 1
}
```

Delete trial

DELETE `/api/trails/<your_trial_id>/`

Return :

none

5.6 Cheat-Sheet

5.6.1 Experiment

Here is an example of an **Experiment** object

```
{
    "algo_count": 1,
    "created": "2018-11-20T09:43:43.442381Z",
    "dataset": "Iris_dataset.csv",
    "dataset_parameters": {"param_1": "somevalue", "param_2": 12},
    "description": "It is just a demo experiment to show you what's inside.",
    "id": "6616120a-28b2-4c8b-a1c6-7f18c639632c",
    "metrics": [
        { "metric_name": "metric_1", "type": "reward" },
        { "metric_name": "metric_2", "type": "loss" }
```

(continues on next page)

(continued from previous page)

```
    ],
    "name": "Demo Experiment",
    "owner": "bender_admin",
    "trial_count": 2
}
```

- **algo_count** : *number* - Number of algos in this experiment
- **created** : *string* - Date of creation
- **dataset** : *string* - Name of the used dataset
- **dataset_parameters** : *dict* - Custom object to store your own metadata about your experiment
- **description** : *string* - Quick description of the experiment
- **id** : *string* - Id of the experiment
- **metrics**: *array* - Each one of them should be either from type `loss` or `reward` depending if you are trying to `minimize` or `maximize` this metric in your experiment.
- **name** : *string* - Name of the experiment
- **owner** : *string* - Owner of the experiment
- **trial_count** : *number* - Number of trials in this experiment

5.6.2 Algo

Here is an example of an **Algo** object

```
{
    "id": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "sklearn-svm",
    "owner": "bender_admin",
    "experiment": "6616120a-28b2-4c8b-a1c6-7f18c639632c",
    "created": "2018-11-07T17:01:27.286194Z",
    "description": "One of the various possibilities.",
    "parameters": [
        {
            "algo": "0597ca48-66f7-42be-9021-12ec57d4251e"
            "category": "categorical"
            "name": "kernel"
            "search_space": { "values": ["linear", "poly", "rbf", "sigmoid"] }
        },
        {
            "algo": "0597ca48-66f7-42be-9021-12ec57d4251e"
            "category": "uniform"
            "name": "C"
            "search_space": { "high": 5, "step": 0.1, "low": -5 }
        }
    ]
    "trial_count": 60
}
```

- **created** : *string* - Date of creation
- **description** : *string* - Quick description of the algo
- **experiment** : *string* - Id of the experiment

- **id** : *string* - Id of the algo
- **name** : *string* - Name of the algo
- **owner** : *string* - Owner of the algo
- **parameters**: *array* - An array of hyperparameters (see the hyperparameters section just below)
- **trial_count** : *number* - Number of trials in this

5.6.3 Hyperparameters

Here are some examples of **Hyperparameters** objects that can appear in an Algo.

```
{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x1",
    "category": "uniform",
    "search_space": {
        "low": 0,
        "high": 10,
    }
}  #  some examples: 8.364, 2.3, 4.5, etc.

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x1_step",
    "category": "uniform",
    "search_space": {
        "low": 0,
        "high": 10,
        "step": 1
    }
}  #  some examples: 0, 5, 6, 7, etc.

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x2",
    "category": "loguniform",
    "search_space": {
        "low": 1e4,
        "high": 1e6,
        "base": 10,
    }
}  #  some examples: 3.14456e4, 5.36412e5, 9.12450e6, etc.

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x2_step",
    "category": "loguniform",
    "search_space": {
        "low": 1e4,
        "high": 1e6,
        "step": 1e3,
        "base": 10,
    }
}  #  some examples: 3.1e4, 5.36e5, 9.126e6, etc.
```

(continues on next page)

(continued from previous page)

```
{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x3",
    "category": "normal",
    "search_space": {
        "mu": 8,
        "sigma": 4,
        "low": 0,
        "high": 10,
    }  #  some examples: 8.3, 7.5, 5.6, 7.9, etc.
}

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x3_step",
    "category": "normal",
    "search_space": {
        "mu": 8,
        "sigma": 4,
        "low": 0,
        "high": 10,
        "step": 0.2,
    }
}  #  some examples: 8.2, 8, 7.6, 5.6, etc.

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x4",
    "category": "lognormal",
    "search_space": {
        "mu": 1e-5,
        "sigma": 1e1,
        "low": 1e-7,
        "high": 1e-3,
        "base": 10,
    }
}  #  some examples: 1.2e-5, 0.3e-6, 7.65e-4 etc.

{
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "name": "x5",
    "category": "categorical",
    "search_space": {
        "values": ["a", "b", "c", "d"],
        "probabilities": [1 / 3, 1 / 3, 1 / 6, 1 / 6]
    }
}  #  some examples: a, b, a, b, c, etc.
```

- **algo**: *string* - Id of the algo
- **name**: *string* - Name of your hyperparameter
- **category**: *string*->[*enum*] - One of the following values depending on the type of your variable : `categorical`, `uniform`, `loguniform`, `normal`, `lognormal`
- **search_space**: *dict* - Depending on the value of the 'category' field, the search_space dict can or must contain different keys. To know what to fill, look at the matching table below.

`step`, *if not specificated, will explore the search space in a continuous interval*

`base`, *is set at 10 by default*

	uniform	loguniform	normal	lognormal	categorical
mu	`forbidden`	`forbidden`	**mandatory**	**mandatory**	`forbidden`
sigma	`forbidden`	`forbidden`	**mandatory**	**mandatory**	`forbidden`
low	**mandatory**	**mandatory**	**mandatory**	**mandatory**	`forbidden`
high	**mandatory**	**mandatory**	**mandatory**	**mandatory**	`forbidden`
step	*optional*	*optional*	*optional*	*optional*	`forbidden`
base	`forbidden`	*optional*	`forbidden`	*optional*	`forbidden`
values	`forbidden`	`forbidden`	`forbidden`	`forbidden`	**mandatory**
probabilities	`forbidden`	`forbidden`	`forbidden`	`forbidden`	*optional*

5.6.4 Trial

Here is an example of an **Trial** object

```
{
    "id": "d188b0e6-9080-415d-be78-57efe8589a80",
    "algo_name": "sklearn-svm",
    "algo": "0597ca48-66f7-42be-9021-12ec57d4251e",
    "comment": "Pretty much nothing",
    "created": "2018-11-07T17:01:27.292336Z",
    "experiment": "6616120a-28b2-4c8b-a1c6-7f18c639632c",
    "owner": "bender_admin",
    "parameters": {
        "C": 0.07699688616826196,
        "kernel": "poly"
    },
    "results": {
        "test_accuracy": 1,
        "test_cohen_kappa": 1,
        "train_accuracy": 0.97,
        "train_cohen_kappa": 0.9546896239238786
    },
    "weight": 1
}
```

- **algo** : *string* - Id of the algo
- **algo_name** : *string* - Name of the algo
- **comment** : *string* - Something to say about this trial
- **created** : *string* - Date of creation
- **experiment** : *string* - Id of the experiment
- **id** : *string* - Id of the trial
- **owner** : *string* - Owner of the experiment
- **parameters** : *dict* - Values of hyperparameters used for this trial
- **results** : *dict* - Metric results for this trial.
- **weight** : *number* - optional Importance of this trial compared to the others (default 1)

5.6.5 Suggestions

The currently supported optimizers are :

`parzen_estimator`

`random`

5.7 Python

PLEASE READ THE 'CONCEPTS' DOCUMENTATION TAB FIRST IF IT'S YOUR FIRST TIME USING THE CLIENT

5.7.1 Install

The **python client for bender** is available with the `pip install bender-client` command.

Or you can just download it from it's github repo if you want.

5.7.2 Import and Login

Make sure you already created a Bender account on the web client !

```
from benderclient import Bender
b = Bender()
```

When you create a new bender object, it will automatically try to connect to the bender API so make sure you have a working internet connection.

The log-in menu will be prompt in the python console and ask for your email and password : it will create a temporary file on your computer with your credentials so you won't have to login each time.

When running the Bender ecosystem, you want to use your own API, so when calling **b = Bender(host='https://bender-api.dreem.com')** you can set a host parameter which is by default set to the adress of our servers.

5.7.3 Experiment methods

List experiments

Return a list of the connected user's experiments.

```
b.list_experiments()
```

Prototype :

`list_experiments()`

Arguments :

none

Return :

array

```
[{ "name":  "exp_1", "id":  "0597ca48-6..." }, { "name":  "exp_2", "id":
"68hj547r5-8..." } ]
```

Set experiment

Setup current experiment for the connected user.

```
b.set_experiment(name, experiment_id)
```

Prototype :

```
set_experiment(name=None, experiment_id=None)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Give the name of the experiment you want to retrieve and set as current
		`"my_experiment"`
experiment_id	*string*	Give the id of the experiment you want to retrieve and set as current
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

Create experiment

Create new experiment and setup current experiment for the connected user.

If experiment name already exists, the already existing one is set by default as current experiment.

```
b.create_experiment(name, metrics, description, dataset, dataset_parameters)
```

Prototype :

```
create_experiment(name, metrics, description=None, dataset=None,
dataset_parameters=None)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Name of your experiment
		`"my_experiment"`
metrics	*array*	List of the performance metrics used to evaluate your experiment
		`[ {"metric_name":  "metric_a","type":  "reward" }, {"metric_name":  "metric_b","type":  "loss" }]`
description	*string*	Short description of the experiment's purpose
		`"This experiment is just a random algorithm."`
dataset	*string*	Name the dataset you are using for your experiment
		`"dataset_name.csv"`
dataset	*dict*	An object describing your dataset
		`{ "version":  0.1, "CV_folds":  "10" }`

Return :

none

Get experiment

Return the current experiment.

```
b.get_experiment()
```

Prototype :

```
get_experiment()
```

Arguments :

none

Return :

dict

A full experiment object.

Delete experiment

Delete targeted experiment of the connected user.

```
b.delete_experiment(experiment_id)
```

Prototype :

```
delete_experiment(experiment_id=None)
```

Arguments :

Argument	Type	Description and Example
experiment_id	*string*	Give the id of the experiment you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.7.4 Algo methods

List algos

Return a list of the connected user's algos.

```
b.list_algos()
```

Prototype :

```
list_algos()
```

Arguments :

none

Return :

array

```
[{ "name":  "algo_1", "id":  "0597ca48-6..." }, { "name":  "algo_2", "id":
"68hj547r5-8..." } ]
```

Set algo

Setup current algo for the connected user.

```
b.set_algo(name, algo_id)
```

Prototype :

```
set_algo(name=None, algo_id=None)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Give the name of the algo you want to retrieve and set as current
		`"my_algo"`
algo_id	*string*	Give the id of the algo you want to retrieve and set as current
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

Create algo

Create new algo and setup current algo for the connected user.

If algo name already exists, the already existing one is set by default as current algo.

```
b.create_algo(name, parameters, description)
```

Prototype :

```
create_algo(name, hyperparameters, description=None)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Name of your algo
		`"my_algo"`
hyperparameters	*array*	List of the hyperparameters used by the algo
		`[{"name":  "param_name","category":  "categorical", "search_space":  { "values":  [3, 5, 7] } }]`
description	*string*	Short description of the algo's principle
		`"This algo is a useless one."`

Return :

none

Get algo

Return the current algo.

```
b.get_algo()
```

Prototype :

`get_algo()`

Arguments :

none

Return :

dict

A full algo object.

Delete algo

Delete targeted algo of the connected user.

```
b.delete_algo(algo_id)
```

Prototype :

`delete_algo(algo_id=None)`

Arguments :

Argument	Type	Description and Example
algo_id	*string*	Give the id of the algo you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.7.5 Trials methods

List trials

List all trials of the current algo.

```
b.list_trials()
```

Prototype :

```
list_trials()
```

Arguments :

none

Return :

array

An array of trials dict.

Create trial

Create new trial for the current algo.

```
b.create_trial(name, hyperparameters, description)
```

Prototype :

```
create_trial(results, hyperparameters, weight=1, comment=None)
```

Arguments :

Argument	Type	Description and Example
results	*dict*	Array of obtained metrics
		`{"metric1": 0.8, "metric2": 0.3}`
hyperparameters	*dict*	List of the hyperparameters used by the algo
		`{"param1": "value","param2": 3567, "param3": "another"}`
weight	*integer*	The importance of your result
		`0.5`
comment	*string*	Anything you want to say about this trial
		`"This is a normal trial."`

Return :

none

Delete trial

Delete targeted trial from current algo.

```
b.delete_trial(trial_id)
```

Prototype :

```
delete_trial(trial_id=None)
```

Arguments :

Argument	Type	Description and Example
trial_id	*string*	Give the id of the trial you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.7.6 General methods

Suggest

Ask bender a suggestion on a hyperparameters set to use

```
b.suggest(metric, optimizer)
```

Prototype :

```
suggest(metric=None, optimizer="parzen_estimator")
```

Arguments :

Return :

dict

```
{"param1":  "value","param2":  3567, "param3":  "another"}
```

Revoke credentials

Remove the registered credentials from this computer

```
b.revoke_credentials()
```

Hello

```
b._say_hello()
```

5.8 R

PLEASE READ THE 'CONCEPTS' DOCUMENTATION TAB FIRST IF IT'S YOUR FIRST TIME USING THE CLIENT

5.8.1 Install

The **r client for bender** is available with the CRAN

Or with the following command within R-Studio `devtools::install_github("Dreem-Organization/bender-r-client")`

5.8.2 Initialization and Login

Make sure you already created a Bender account on the web client !

```
b = bender::Bender$new("my_email", "my_password", host='https://bender-api.dreem.com')
```

When you create a new bender object, it will automatically try to connect to the bender API so make sure you have a working internet connection.

The email and password are the one you use to log into bender.dreem.com

When running the Bender ecosystem, you want to use your own API, so when initializing the bender object you can set a host parameter which is by default set to the adress of our servers.

5.8.3 Experiment methods

List experiments

Return a list of the connected user's experiments.

```
b$list_experiments()
```

Prototype :

```
list_experiments()
```

Arguments :

none

Return :

list

```
list(list(name="exp_1", id="0597ca48-6..."), list(name="exp_2",
id="68hj547r5-6..."))
```

Set experiment

Setup current experiment for the connected user.

```
b$set_experiment(name, id)
```

Prototype :

```
set_experiment(name=NULL, id=NULL)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Give the name of the experiment you want to retrieve and set as current
		`"my_experiment"`
id	*string*	Give the id of the experiment you want to retrieve and set as current
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

Create experiment

Create new experiment and setup current experiment for the connected user.

If experiment name already exists, the already existing one is set by default as current experiment.

```
b$create_experiment(name, metrics, description, dataset, dataset_parameters)
```

Prototype :

```
create_experiment(name, metrics, description=NULL, dataset=NULL,
dataset_parameters=NULL)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Name of your experiment
		`"my_experiment"`
metrics	*list*	List of the performance metrics used to evaluate your experiment
		`list(list(metric_name="metric_a", type="reward" ), list(metric_name="metric_b", type="loss"))`
description	*string*	Short description of the experiment's purpose
		`"This experiment is just a random algorithm."`
dataset	*string*	Name the dataset you are using for your experiment
		`"dataset_name.csv"`
dataset	*list*	An object describing your dataset
		`list(version=0.1, CV_folds="10")`

Return :

none

Delete experiment

Delete targeted experiment of the connected user.

```
b$delete_experiment(id)
```

Prototype :

```
delete_experiment(id=NULL)
```

Arguments :

Argument	Type	Description and Example
id	*string*	Give the id of the experiment you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.8.4 Algo methods

List algos

Return a list of the connected user's algos.

```
b$list_algos()
```

Prototype :

```
list_algos()
```

Arguments :

none

Return :

list

```
list(list(name="algo_1", id="0597ca48-6..."), list(name="algo_2",
id="68hj547r5-6..."))
```

Set algo

Setup current algo for the connected user.

```
b$set_algo(name, id)
```

Prototype :

```
set_algo(name=NULL, id=NULL)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Give the name of the algo you want to retrieve and set as current
		`"my_algo"`
id	*string*	Give the id of the algo you want to retrieve and set as current
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

Create algo

Create new algo and setup current algo for the connected user.

If algo name already exists, the already existing one is set by default as current algo.

```
b$create_algo(name, hyperparameters, description)
```

Prototype :

```
create_algo(name, hyperparameters, description=NULL)
```

Arguments :

Argument	Type	Description and Example
name	*string*	Name of your algo
		`"my_algo"`
hyperparameters	*list*	List of the hyperparameters used by the algo
		`list(list(name="param_name",category="categorical", search_space=list(values=list(3, 5, 7))))`
description	*string*	Short description of the algo's principle
		`"This algo is a useless one."`

Return :

none

Delete algo

Delete targeted algo of the connected user.

```
b.delete_algo(algo_id)
```

Prototype :

```
delete_algo(algo_id=NULL)
```

Arguments :

Argument	Type	Description and Example
algo_id	*string*	Give the id of the algo you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.8.5 Trials methods

List trials

List all trials of the current algo.

```
b$list_trials()
```

Prototype :

```
list_trials()
```

Arguments :

none

Return :

list

A list of trials.

Create trial

Create new trial for the current algo.

```
b$create_trial(name, hyperparameters, description)
```

Prototype :

```
create_trial(results, hyperparameters, weight=1, comment=NULL)
```

Arguments :

Argument	Type	Description and Example
results	*list*	Array of obtained metrics
		`list(metric1=0.8, metric2=0.3)`
hyperparameters	*list*	List of the hyperparameters used by the algo
		`list(param1="value", param2=3567, param3="another"}`
weight	*integer*	The importance of your result
		`0.5`
comment	*string*	Anything you want to say about this trial
		`"This is a normal trial."`

Return :

none

Delete trial

Delete targeted trial from current algo.

```
b$delete_trial(trial_id)
```

Prototype :

```
delete_trial(id=NULL)
```

Arguments :

Argument	Type	Description and Example
id	*string*	Give the id of the trial you want to retrieve and delete
		`"0597ca48-66f7-42be-9021-12ec57d4251e"`

Return :

none

5.8.6 General methods

Suggest

Ask bender a suggestion on a hyperparameters set to use

```
b$suggest(metric, optimizer)
```

Prototype :

```
suggest(metric=NULL, optimizer="parzen_estimator")
```

Arguments :

Return :

list

```
list(param1="value", param2=3567, param3="another")
```