hyperpy

Release 0.0.4

Sergio A. Mora Pardo

Dec 20, 2021

GETTING STARTED

hyperpy

HyperPy: An automatic hyperparameter optimization framework

HyperPy (py-hyperpy in PyPi) is a Python library for build an automatic hyperparameter optimization.

You can install *hyperpy* with pip:

```
# pip install py-hyperpy
```

CHAPTER

ONE

EXAMPLE

import library:

```
import hyperpy as hy
```

Run the optimization:

```
running=hy.run(feat_X, Y)
study = running.buildStudy()
```

See the results:

```
print("best params: ", study.best_params)
print("best test accuracy: ", study.best_value)
best_params, best_value = hy.results.results(study)
```

Note:

- The function *hy.run()* return a *Study* object. And only needs: Features, target. In the example: best test accuracy = 0.7407407164573669
- *feat_X*: features in dataset
- *Y*: target in dataset

Warning: At moment only solves binary clasification problems.

Note: This project is active development.

Citing HyperPy:

If you're citing HyperPy in research or scientific paper, please cite this page as the resource. HyperPy's first stable release 0.0.5 was made publicly available in October 2021. py-hyperpy.readthedocs. HyperPy, October 2021. URL https://py-hyperpy.readthedocs.io/en/latest/. HyperPy version 0.0.5.

A formatted version of the citation would look like this:

```
@Manual{HyperPy,
  author  = {Mora, Sergio},
  title   = {HyperPy: An automatic hyperparameter optimization framework in Python},
```

(continues on next page)

(continued from previous page)

```
  year    = {2021},
  month   = {October},
  note    = {HyperPy version 0.0.5},
  url     = {https://py-hyperpy.readthedocs.io/en/latest/}
}
```

We are appreciated that HyperPy has been increasingly referred and cited in scientific works. See all citations here: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=hyperpy&btnG=

Key Links and Resources:

- Release Notes
- Example Notebooks
- Blog Posts
- Contribute
- More about HyperPy

1.1 Install

1.1.1 Installation

HyperPy (py-hyperpy in PyPi) is a Python library for build an automatic hyperparameter optimization.

You can install *hyperpy* with pip:

```
(.venv) $ pip install py-hyperpy
```

1.1.2 Example

import library:

```
import hyperpy as hy
```

Run the optimization:

```
running=hy.run(feat_X, Y)
study = running.buildStudy()
```

See the results:

```
print("best params: ", study.best_params)
print("best test accuracy: ", study.best_value)
best_params, best_value = hy.results.results(study)
```

Note:

- The function *hy.run()* return a *Study* object. And only needs: Features, target. In the example: best test accuracy = 0.7407407164573669
- *feat_X*: features in dataset

- *Y*: target in dataset

Warning: At moment only solves binary clasification problems.

Note: This project is active development.

1.2 Usage

1.2.1 Installation

To use Py-Hyperpy, first install it using pip:

```
(.venv) $ pip install py-hyperpy
```

1.2.2 Create Study

Fisrt of all, you need to import library:

```
(.venv) $ import hyperpy as hy
```

The library **hyperpy** function by study. This study represent several running of an diferents neural networks, to find the best fit. To run a study, you could call `hy.run(feat_X, Y)` function:

class hyperpy.core.**run**(*feat_X*, *Y*, *study_name: str = 'First try'*, *direction: str = 'maximize'*, *n_trials: int = 10*)
run class is used to run the experiment.

objective(*trial*)
objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

buildStudy()
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

hyperpy.core.run.**buildStudy**(*self*)
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

The `Feat_X` parameter should be the feature to train the model. And `Y` represents the target in dataset. However, `hy.run()` at the moment just run clasification problems and run study with doble cross validation.

For example:

```
>>> import hyperpy as hy
>>> running=hy.run(feat_X, Y)
>>> study = running.buildStudy()
```

Then the study return the structure of the neural netowork and the accuracy.

1.3 Classes

1.3.1 Class models

The class *models* builds a model from a set of parameters.

class hyperpy.core.**models**(*initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>, min_layers: int = 1, max_layers: int = 13, min_units: int = 4, max_units: int = 128*)

Class to build a model with a given topology

BuildModelSimply(*self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModelSimply Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model.Sequential

BuildModel(*self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModel Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model

The fact, all parameters for build model are (default):

- initnorm=keras.initializers.RandomNormal(mean=0.0, stddev=0.05, seed=1),
- min_layers:int=1,
- max_layers:int=13,
- min_units:int=4,
- max_units:int=128

and at the moment we can manipulate the model with the following methods:

hyperpy.core.models.**BuildModelSimply**(*trial: <Mock name='mock.Trial' id='140024184256400'>, self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModelSimply Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model.Sequential

hyperpy.core.models.**BuildModel**(*trial: <Mock name='mock.Trial' id='140024184256400'>, self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModel Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model

The difference between th two methods is the first use the same activation function for all layers, the second use different activations funcion for each layer.

1.3.2 Class optimizers

The class *optimizers* build optimizers for the model.

class hyperpy.core.**optimizers**
class to build a model optimizer

optimizerAdam() → <Mock name='mock.optimizers.Adam' id='140024185367504'>
optimizerAdam method to build a model optimizer with Adam

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.Adam

optimizerRMSprop() → <Mock name='mock.optimizers.RMSprop' id='140024185110672'>
optimizerRMSprop method to build a model optimizer with RMSprop

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.RMSprop

optimizerSGD() → <Mock name='mock.optimizers.SGD' id='140024185112912'>
optimizerSGD method to build a model optimizer with SGD

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.SGD

buildOptimizer() → None
buildOptimizer method to build a model optimizer

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers

At the moment, we can select between:

hyperpy.core.optimizers.**optimizerAdam**(*trial: <Mock name='mock.Trial' id='140024184256400'>*) → <Mock name='mock.optimizers.Adam' id='140024185367504'>
optimizerAdam method to build a model optimizer with Adam

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.Adam

hyperpy.core.optimizers.**optimizerRMSprop**(*trial: <Mock name='mock.Trial' id='140024184256400'>*) → <Mock name='mock.optimizers.RMSprop' id='140024185110672'>

optimizerRMSprop method to build a model optimizer with RMSprop

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.RMSprop

hyperpy.core.optimizers.**optimizerSGD**(*trial: <Mock name='mock.Trial' id='140024184256400'>*) → <Mock name='mock.optimizers.SGD' id='140024185112912'>

optimizerSGD method to build a model optimizer with SGD

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.SGD

And if we want that the model is trained with several optimizers, we can use the method:

hyperpy.core.optimizers.**buildOptimizer**(*trial: <Mock name='mock.Trial' id='140024184256400'>*) → None

buildOptimizer method to build a model optimizer

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers

1.3.3 Class trainers

The class *trainers* build trainers for the model.

class hyperpy.core.**trainers**(*trial*, *feat_X*, *Y*, *verbose: int = 0*, *model: hyperpy.core.models = <class 'hyperpy.core.models'>*, *optimizer: hyperpy.core.optimizers = <class 'hyperpy.core.optimizers'>*, *type: str = 'Build'*, *initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*)

trainers class to build a model trainer

trainer(*save: bool = False*) → None

trainer trainer Method define how to train Neural Network. This works by maximizing the test data set (Exactitud de Validación).

Parameters **save** (*bool, optional*) – save model, defaults to False

Returns model, cv_x, cv_y

Return type keras.models, pandas.DataFrame, pandas.Series

The final idea, is to select by several type of trainers. By the way, at moment have onle one trainer:

hyperpy.core.trainers.**trainer**(*self*, *save: bool = False*) → None

trainer trainer Method define how to train Neural Network. This works by maximizing the test data set (Exactitud de Validación).

Parameters **save** (*bool, optional*) – save model, defaults to False

Returns model, cv_x, cv_y

Return type keras.models, pandas.DataFrame, pandas.Series

1.3.4 Class run

To run a study, you could call `hy.run(feat_X, Y)` function:

class hyperpy.core.**run**(*feat_X*, *Y*, *study_name: str = 'First try'*, *direction: str = 'maximize'*, *n_trials: int = 10*)
run class is used to run the experiment.

objective(*trial*)
objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

buildStudy()
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

hyperpy.core.run.**buildStudy**(*self*)
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

hyperpy.core.run.**objective**(*self*, *trial*)
objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

1.3.5 Class results

To read results from a study, you could call `hy.results(study)` function:

class hyperpy.core.**results**
results class is used to get the results of the study.

results()
results function is used to get the results of the study.

Parameters **study** (*optuna.study.Study*) – study object

Returns results

Return type pandas.DataFrame

hyperpy.core.results.**results**(*study*)
results function is used to get the results of the study.

Parameters **study** (*optuna.study.Study*) – study object

Returns results

Return type pandas.DataFrame

1.4 Modules

1.4.1 Core

class hyperpy.core.**models**(*initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*, *min_layers: int = 1*, *max_layers: int = 13*, *min_units: int = 4*, *max_units: int = 128*)

Class to build a model with a given topology

BuildModelSimply(*self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModelSimply Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model.Sequential

BuildModel(*self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModel Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model

class hyperpy.core.**optimizers**

class to build a model optimizer

optimizerAdam() → <Mock name='mock.optimizers.Adam' id='140024185367504'>

optimizerAdam method to build a model optimizer with Adam

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.Adam

optimizerRMSprop() → <Mock name='mock.optimizers.RMSprop' id='140024185110672'>

optimizerRMSprop method to build a model optimizer with RMSprop

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.RMSprop

optimizerSGD() → <Mock name='mock.optimizers.SGD' id='140024185112912'>

optimizerSGD method to build a model optimizer with SGD

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.SGD

buildOptimizer() → None

buildOptimizer method to build a model optimizer

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers

class hyperpy.core.**trainers**(*trial*, *feat_X*, *Y*, *verbose: int = 0*, *model: hyperpy.core.models = <class 'hyperpy.core.models'>*, *optimizer: hyperpy.core.optimizers = <class 'hyperpy.core.optimizers'>*, *type: str = 'Build'*, *initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*)

trainers class to build a model trainer

trainer(*save: bool = False*) → None

trainer trainer Method define how to train Neural Network. This works by maximizing the test data set (Exactitud de Validación).

Parameters **save** (*bool, optional*) – save model, defaults to False

Returns model, cv_x, cv_y

Return type keras.models, pandas.DataFrame, pandas.Series

class hyperpy.core.**run**(*feat_X*, *Y*, *study_name: str = 'First try'*, *direction: str = 'maximize'*, *n_trials: int = 10*)

run class is used to run the experiment.

objective(*trial*)

objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

buildStudy()

buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

class hyperpy.core.**results**

results class is used to get the results of the study.

results()

results function is used to get the results of the study.

Parameters **study** (*optuna.study.Study*) – study object

Returns results

Return type pandas.DataFrame

1.4.2 Utils

1.5 Classification

class hyperpy.core.**models**(*initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*, *min_layers: int = 1*, *max_layers: int = 13*, *min_units: int = 4*, *max_units: int = 128*)

Class to build a model with a given topology

BuildModelSimply(*self*) → <Mock name='mock.models.Model' id='140024185113744'>

BuildModelSimply Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model.Sequential

BuildModel(*self*) → <Mock name='mock.models.Model' id='140024185113744'>
BuildModel Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model

class hyperpy.core.**optimizers**
class to build a model optimizer

optimizerAdam() → <Mock name='mock.optimizers.Adam' id='140024185367504'>
optimizerAdam method to build a model optimizer with Adam

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.Adam

optimizerRMSprop() → <Mock name='mock.optimizers.RMSprop' id='140024185110672'>
optimizerRMSprop method to build a model optimizer with RMSprop

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.RMSprop

optimizerSGD() → <Mock name='mock.optimizers.SGD' id='140024185112912'>
optimizerSGD method to build a model optimizer with SGD

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.SGD

buildOptimizer() → None
buildOptimizer method to build a model optimizer

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers

class hyperpy.core.**trainers**(*trial*, *feat_X*, *Y*, *verbose: int = 0*, *model: hyperpy.core.models = <class 'hyperpy.core.models'>*, *optimizer: hyperpy.core.optimizers = <class 'hyperpy.core.optimizers'>*, *type: str = 'Build'*, *initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*)
trainers class to build a model trainer

trainer(*save: bool = False*) → None
trainer trainer Method define how to train Neural Network. This works by maximizing the test data set (Exactitud de Validación).

Parameters **save** (*bool, optional*) – save model, defaults to False

Returns model, cv_x, cv_y

Return type keras.models, pandas.DataFrame, pandas.Series

class hyperpy.core.**run**(*feat_X*, *Y*, *study_name: str = 'First try'*, *direction: str = 'maximize'*, *n_trials: int = 10*)
run class is used to run the experiment.

objective(*trial*)
objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

buildStudy()
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

class hyperpy.core.**results**
results class is used to get the results of the study.

results()
results function is used to get the results of the study.

Parameters **study** (*optuna.study.Study*) – study object

Returns results

Return type pandas.DataFrame

1.6 Regression

class hyperpy.core.**models**(*initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*, *min_layers: int = 1*, *max_layers: int = 13*, *min_units: int = 4*, *max_units: int = 128*)
Class to build a model with a given topology

BuildModelSimply(*self*) → <Mock name='mock.models.Model' id='140024185113744'>
BuildModelSimply Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model.Sequential

BuildModel(*self*) → <Mock name='mock.models.Model' id='140024185113744'>
BuildModel Standar model

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns sequential model

Return type keras.models.Model

class hyperpy.core.**optimizers**
class to build a model optimizer

optimizerAdam() → <Mock name='mock.optimizers.Adam' id='140024185367504'>
optimizerAdam method to build a model optimizer with Adam

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.Adam

optimizerRMSprop() → <Mock name='mock.optimizers.RMSprop' id='140024185110672'>
optimizerRMSprop method to build a model optimizer with RMSprop

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.RMSprop

optimizerSGD() → <Mock name='mock.optimizers.SGD' id='140024185112912'>
optimizerSGD method to build a model optimizer with SGD

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers.SGD

buildOptimizer() → None
buildOptimizer method to build a model optimizer

Parameters **trial** (*optuna.Trial*) – trial to build the model

Returns optimizer

Return type keras.optimizers

class hyperpy.core.**trainers**(*trial*, *feat_X*, *Y*, *verbose: int = 0*, *model: hyperpy.core.models = <class 'hyperpy.core.models'>*, *optimizer: hyperpy.core.optimizers = <class 'hyperpy.core.optimizers'>*, *type: str = 'Build'*, *initnorm=<Mock name='mock.initializers.RandomNormal()' id='140024184255760'>*)
trainers class to build a model trainer

trainer(*save: bool = False*) → None
trainer trainer Method define how to train Neural Network. This works by maximizing the test data set (Exactitud de Validación).

Parameters **save** (*bool, optional*) – save model, defaults to False

Returns model, cv_x, cv_y

Return type keras.models, pandas.DataFrame, pandas.Series

class hyperpy.core.**run**(*feat_X*, *Y*, *study_name: str = 'First try'*, *direction: str = 'maximize'*, *n_trials: int = 10*)
run class is used to run the experiment.

objective(*trial*)
objective function is used to define the objective function.

Parameters **trial** (*optuna.trial.Trial*) – trial object

Returns objective function

Return type float

buildStudy()
buildStudy function is used to build the study.

Returns study

Return type optuna.study.Study

class hyperpy.core.**results**

results class is used to get the results of the study.

results()

results function is used to get the results of the study.

Parameters **study** (*optuna.study.Study*) – study object

Returns results

Return type pandas.DataFrame

PYTHON MODULE INDEX

h

INDEX