NN for CHF prediction

Dumont Jules

Oct 06, 2023

CONTENTS:

1 CHF_model_api **1**
1.1 CHF_model_api package 1

2 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER

ONE

CHF_MODEL_API

1.1 CHF_model_api package

1.1.1 Submodules

1.1.2 CHF_model_api.config module

1.1.3 CHF_model_api.dataBase module

class CHF_model_api.dataBase.**MyDB**(*seed: int = 1*, *input_number: int = 4*, *interpolation: bool = False*)

Bases: `object`

This class purpose is to make object containing the data needed for training and validation and making data quickly accessible. It can be vizualise as an object representation of the LUT table, it will contain linear interpolation function of the LUT table

AVAILABLE_DB = []

ISUnitsTransformation() → DataFrame

Convert the raw data in SI units and return

extractSortFromPdf(*path*) → DataFrame

create a csv file based on Groeneveld 2006 LUT pdf take 2 min to run

extraction(*path*) → None

extract the data from pdf and put it in original_data.csv take 2 min to run

filtration(*sort: DataFrame*) → DataFrame

The dataframe is filtered in order to kick outliers and nonsense values

classmethod getAvailableDataBases() → List[*MyDB*]

getInterpFunction() → LinearNDInterpolator

return the linear interpolator object/function

getLUTPerformances() → None

print lut performances

interpolate(*X_list*) → float

allow to use the LUT interpolated function input : [LD,P,G,Xchf]

isCompatible(*model:* MyModel) → bool

return True if the database can be used for training a model based

loadData(*data_seed: int = 1*, *input_number: int = 5*) → dict

take the data from a csv containing data in IS units or create it from the Groeneveld 2006 LUT pdf return a dict containing the keys: 'validation_targets', 'validation_features','training_features' 'training_targets', 'mean' 'std'(mean and std of the training_features before normalization we need to keep when predicting) and is meant to be add to DATA[seed] = {'validation':...}

makeDictDatabase(*input_number: int*, *training_data: DataFrame*, *validation_data: DataFrame*) → dict

select the desired data features tu put it in a dict

normalizeData(*data: dict*) → dict

take the dict with the data and return it normalized

1.1.4 CHF_model_api.model module

class CHF_model_api.model.**MyModel**(*hparams: dict | None = None*, *model_name: str | None = None*, *auto_save: bool = False*, *process_number: int | None = None*)

Bases: object

class that either create a CHF prediction model from scratch using hparams dictionnary or load a saved one from model_name from a .h5 file and his hp from json

createNewModel() → None

create new model based on hparams

displayPerf()

init_callbacks() → list

action taken at evry epoch for monitoring

loadData() → None

check if corresponding valid/train set already computed, if no compute and add the new data in DATA

loadMyModel(*model_name: str*) → None

Create a model based on hparams given in parameterss or use a saved one based on his name and load it from saved_models directory set also all the attributes

lrScheduler(*epoch: int*, *lr: float*) → float

program a decrease of the learning rate

makeRealPredictions(*features_data: list*) → list

make CHF predicitons

plotLoss(*save=False*) → None

Plot the loss and validation loss to see if overfitting

plotResult() → None

save(*overwrite=False*) → None

save model in .h5 format, to make predictions save also the hparams of the model in json files

saveResults() → None

compute and save metrics in hparams

train(*logs=True*, *callbacks=True*, *train_epochs=None*) → dict

train the model over train_epochs epochs and save the results in the attribute hparams

vizualize() → None

save a png file containing vizual rpz of the network

CHF_model_api.model.**batch_nrmse**(*y_true*, *y_pred*) → float

1.1.5 CHF_model_api.optimizer module

class CHF_model_api.optimizer.**MyOptimizer**(*generic_hparams: dict*, *opti_architecture: bool = True*, *opti_dropout: bool = False*, *opti_learning_rate: bool = False*, *opti_lr_decrease: bool = False*, *opti_optimizer: bool = False*, *opti_activation_method: bool = False*, *type: int = 1*, *jobs: int = 2*, *trials: int = 5*, *db_name: str | None = None*, *metric: str = 'mpe'*, *verbose: int = 0*)

Bases: object

This class create an optimizer for hyperparameter of deep neural network to predict CHF using mainly optuna library

access_data_index_safe() → list

make_archi_type(*trial*, *type*, *opti_hparams*) → list

make_around_archi(*trial*, *opti_hparams*)

make_decreased_archi(*trial*) → list

make_increased_archi(*trial*) → list

make_random_archi(*trial*) → list

objective(*trial*)

opti_hparams_safe(*trial*) → dict

Return safely the guess optimized of , the hparameters and the epochs number

optimize_my_models()

train_and_report(*model*, *trial*) → None

1.1.6 CHF_model_api.tensorBoard module

class CHF_model_api.tensorBoard.**MyTensorboard**(*saved_models_name: List[str]*)

Bases: object

add_hparams()

add_model(*model*) → None

add_saved_model(*model_name: str*) → None

copy_log(*name*) → None

copy logs in ./logs/modelname/train et /validation to the same directories in ./hparams_tuning/sessionname

copy_logs() → None

get the logs of all the models to compare

init_models() → List[*MyModel*]
get the saved models by creating object my_models based on the name of the saved models

load_tb() → None

log_Hparams() → None

log_Hparams_range() → None

make_Hparams_range() → None

set_up_directories() → None
create validation and training directory and also a directory for each different model (with diff hp)

1.1.7 CHF_model_api.tools module

CHF_model_api.tools.**RemoveDirectoryContent**(*directory_path*) → None
just a function to clean when we want to reset

CHF_model_api.tools.**eraseHparams**(*model_name: str*) → None
Erase stored hyperparameters

CHF_model_api.tools.**getHparamsSavedModel**(*model_name: str*) → dict
return a dict with all the hyperparameters that were saved in the directory hparams in the parent directory saved_models

CHF_model_api.tools.**myMsle**(*y_true*, *y_pred*) → float
Compute mean squared logaritmic error

CHF_model_api.tools.**nrmse**(*y_true*, *y_pred*) → float
Compute normalised root mean suqared error

CHF_model_api.tools.**plotResults**(*predictions: list*, *y_val: list*, *save_fig: bool = False*) → None
Plot the the graph of the predicted values in function of the measured values

CHF_model_api.tools.**remove_backups**(*name*) → None
remove all the saved models .h5 and hparams dict

CHF_model_api.tools.**reset_directories**() → None

CHF_model_api.tools.**saveHparams**(*model_name: str*, *hparams: dict*) → None
save the dict containing results and hyperparameters in hparams idrectory

CHF_model_api.tools.**stdMP**(*y_val*, *predictions*) → float
compute the standart deviation of the ration Measured / predict from 1

CHF_model_api.tools.**testTools**() → bool
print 'package working' if the package is working

CHF_model_api.tools.**utilsNnConfig**(*model*) → List[dict]
used in the visualizeNn function to return the configs of the model

CHF_model_api.tools.**visualizeNn**(*model*, *name*, *description=False*, *figsize=(10, 8)*) → None
Plot the structure of a keras neural network. visualize_nn(model, description=True, figsize=(100,100)) for usage

1.1.8 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

CHF_model_api, 5
CHF_model_api.config, 1
CHF_model_api.dataBase, 1
CHF_model_api.model, 2
CHF_model_api.optimizer, 3
CHF_model_api.tensorBoard, 3
CHF_model_api.tools, 4

INDEX

A

access_data_index_safe() (*CHF_model_api.optimizer.MyOptimizer method*), 3
add_hparams() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
add_model() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
add_saved_model() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
AVAILABLE_DB (*CHF_model_api.dataBase.MyDB attribute*), 1

B

batch_nrmse() (*in module CHF_model_api.model*), 3

C

CHF_model_api
    module, 5
CHF_model_api.config
    module, 1
CHF_model_api.dataBase
    module, 1
CHF_model_api.model
    module, 2
CHF_model_api.optimizer
    module, 3
CHF_model_api.tensorBoard
    module, 3
CHF_model_api.tools
    module, 4
copy_log() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
copy_logs() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
createNewModel() (*CHF_model_api.model.MyModel method*), 2

D

displayPerf() (*CHF_model_api.model.MyModel method*), 2

E

eraseHparams() (*in module CHF_model_api.tools*), 4
extraction() (*CHF_model_api.dataBase.MyDB method*), 1
extractSortFromPdf() (*CHF_model_api.dataBase.MyDB method*), 1

F

filtration() (*CHF_model_api.dataBase.MyDB method*), 1

G

getAvailableDataBases() (*CHF_model_api.dataBase.MyDB class method*), 1
getHparamsSavedModel() (*in module CHF_model_api.tools*), 4
getInterpFunction() (*CHF_model_api.dataBase.MyDB method*), 1
getLUTPerformances() (*CHF_model_api.dataBase.MyDB method*), 1

I

init_callbacks() (*CHF_model_api.model.MyModel method*), 2
init_models() (*CHF_model_api.tensorBoard.MyTensorboard method*), 3
interpolate() (*CHF_model_api.dataBase.MyDB method*), 1
isCompatible() (*CHF_model_api.dataBase.MyDB method*), 1
ISUnitsTransformation() (*CHF_model_api.dataBase.MyDB method*), 1

L

load_tb() (*CHF_model_api.tensorBoard.MyTensorboard method*), 4
loadData() (*CHF_model_api.dataBase.MyDB method*), 1
loadData() (*CHF_model_api.model.MyModel method*), 2

loadMyModel() (*CHF_model_api.model.MyModel method*), 2
log_Hparams() (*CHF_model_api.tensorBoard.MyTensorboard method*), 4
log_Hparams_range() (*CHF_model_api.tensorBoard.MyTensorboard method*), 4
lrScheduler() (*CHF_model_api.model.MyModel method*), 2

M

make_archi_type() (*CHF_model_api.optimizer.MyOptimizer method*), 3
make_around_archi() (*CHF_model_api.optimizer.MyOptimizer method*), 3
make_decreased_archi() (*CHF_model_api.optimizer.MyOptimizer method*), 3
make_Hparams_range() (*CHF_model_api.tensorBoard.MyTensorboard method*), 4
make_increased_archi() (*CHF_model_api.optimizer.MyOptimizer method*), 3
make_random_archi() (*CHF_model_api.optimizer.MyOptimizer method*), 3
makeDictDatabase() (*CHF_model_api.dataBase.MyDB method*), 2
makeRealPredictions() (*CHF_model_api.model.MyModel method*), 2
module
    CHF_model_api, 5
    CHF_model_api.config, 1
    CHF_model_api.dataBase, 1
    CHF_model_api.model, 2
    CHF_model_api.optimizer, 3
    CHF_model_api.tensorBoard, 3
    CHF_model_api.tools, 4
MyDB (*class in CHF_model_api.dataBase*), 1
MyModel (*class in CHF_model_api.model*), 2
myMsle() (*in module CHF_model_api.tools*), 4
MyOptimizer (*class in CHF_model_api.optimizer*), 3
MyTensorboard (*class in CHF_model_api.tensorBoard*), 3

N

normalizeData() (*CHF_model_api.dataBase.MyDB method*), 2
nrmse() (*in module CHF_model_api.tools*), 4

O

objective() (*CHF_model_api.optimizer.MyOptimizer method*), 3
opti_hparams_safe() (*CHF_model_api.optimizer.MyOptimizer method*), 3
optimize_my_models() (*CHF_model_api.optimizer.MyOptimizer method*), 3

P

plotLoss() (*CHF_model_api.model.MyModel method*), 2
plotResult() (*CHF_model_api.model.MyModel method*), 2
plotResults() (*in module CHF_model_api.tools*), 4

R

remove_backups() (*in module CHF_model_api.tools*), 4
RemoveDirectoryContent() (*in module CHF_model_api.tools*), 4
reset_directories() (*in module CHF_model_api.tools*), 4

S

save() (*CHF_model_api.model.MyModel method*), 2
saveHparams() (*in module CHF_model_api.tools*), 4
saveResults() (*CHF_model_api.model.MyModel method*), 2
set_up_directories() (*CHF_model_api.tensorBoard.MyTensorboard method*), 4
stdMP() (*in module CHF_model_api.tools*), 4

T

testTools() (*in module CHF_model_api.tools*), 4
train() (*CHF_model_api.model.MyModel method*), 2
train_and_report() (*CHF_model_api.optimizer.MyOptimizer method*), 3

U

utilsNnConfig() (*in module CHF_model_api.tools*), 4

V

visualizeNn() (*in module CHF_model_api.tools*), 4
vizualize() (*CHF_model_api.model.MyModel method*), 2