RS-HDMR-GPR Code Manual

Version 0.0.1

Owen Ren, Mohamed Boussaidi Ali, Sergei Manzhos

Last Updated: February 8, 2021

1 Installation Guide

The latest version of the code can be obtained from github. The code comes with a IPython jupyter notebook (examples.ipynb) allows one to run and test the examples provided in the paper.

The **rshdmrgpr** package is installed as a Python package. Please install the 64-bit Python version if you are using a Windows machine. We also recommend the user install anaconda distribution to work with virtual environments and conda virtual environments. In this manual we provide a guide for windows installation because all the work done are on Windows. However, there should not be anything that prevents a user from using the package on other operating systems (OS).

1.1 Windows Installation

On windows command prompt (or Anaconda prompt), create a virtual environment and install the package in a directory of your choosing.

```
C:\Users\new_user>python -m env env_name
C:\Users\new_user>env_name\Scripts\activate.bat
(env_name) C:\Users\new_user>pip install git+https://github.com/owen-
   ren0003/rshdmrgpr.git
```

After the package is installed, the user should be able to use it in terminal or any Python IDE with the correct environment setup.

```
(env_name) C:\Users\new_user>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
   :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>  from rshdmrgpr import *
```

Users should consult pip and creating virtualenv for more detailed instructions and commands.

1.2 Anaconda Setup

Once Anaconda is installed, create a conda environment:

```
conda create -n env_name python=3.6
```

Activate the environment, which in this case is called 'env_name':

```
conda activate env_name
```

Install the rshdmrgpr package inside the conda environment:

```
pip install git+https://github.com/owen-ren0003/rshdmrgpr.git
```

Install jupyter lab:

```
conda install -c conda-forge jupyterlab
```

Finally, start jupyterlab to starting working interactively in a jupyter notebook:

```
jupyter lab
```

2 Python API

The code consists of two functions, followed by two classes. The first function is used for to conveniently plot predicted outcomes. The second helper function is used to generate standard unit matrices for hdmr along with isotropic RBF (Radial Basis Function) kernels. The first class creates objects that represent rs-hdmr-gpr models and are used to train and predict on data. Using the trained models from the first class, the second class is utilized to impute missing values. As in the paper, the second class can only impute rows with exactly one missing value.

2.1 Helper function kernel_matrices

rshdmrgpr.**kernel_matrices**(*order, dim, length_scale*)

This function is used to create the RBF kernels and the matrix input for HDMR.

Parameters

- **order:** int – The order of HDMR to use.
- **dim:** int – The dimension of the feature space.
- **length_scale:** float – The length scale to use for the HDMR RBF kernels. (**Note**: If the user would like to create anisotropic kernels it must be done separately. This function only creates RBF isotropic kernels for the corresponding matrices).

Returns

- list of 2D-numpy Array – List of matrices, used to select the features to train the component functions. Has size (dim choose order).
- list of RBF kernels – List of RBF kernels to use for training component function. Has size (dim choose order).

2.2 RSHDMRGPR class

class rshdmrgpr.**RSHDMRGPR**(*num_models, matrices, kernels*)

Bases: object

This class initializes the RS-HDMR-GPR model.

Parameters

- **num_models:** str – The number of HDMR component functions to train.
- **matrices:** list of 2D numpy Array – The matrices that define the HDMR component functions. Every matrix must have the same number of rows equaling the dimension of the feature space.
- **kernels:** list of kernels – The list of sklearn.guassian_process.kernels to use for each HDMR component function.

2.2.1 RSHDMRGPR class methods

RSHDMRGPR.**train**(*self, x_train, y_train, alphas=1e-7, scale_down=(0.5, 1), cycles=50, optimizer=None, opt_every=5,use_columns=None, n_restarts=3, initializer='even', report_rmse=False*)

Trains the component functions for the HDMR model.

Parameters

- **x_train:** pandas DataFrame – The DataFrame containing features.
- **y_train:** pandas Series – The Series containing the target values.

- **alphas:** int **or (**list **of** int**)** – The noise level to be set. If int, the noise level is the set to this for all cycles. To set different noise levels for different cycles, a list of int must be specified.
- **scale_down:** tuple – Must be a tuple, say (s, e) of size two. Training predictions for each component function will be multiplied by $\min\left\{s+\frac{(1-s)ec}{T}, 1\right\}$ on cycle c, where T denotes the total number of cycles.
- **cycles:** int – The number of self consistent cycles to use for training.
- **optimizer:** str **or (**list **of** str**)** – Must be a GPR optimizer or list of such. Please see the sklearn documentation for GaussianProcessRegressor optimizer.
- **opt_every:** int – Specifies every (opt_every) cycles to apply optimizer. Default=1 (optimizer applied every cycle). No difference if optimizer is None.
- **use_columns:** list **of** bool **or None** – Specifies which column to use (True indicates use, false otherwise). If None (the default), all columns are used.
- **n_restarts:** int – Positive integer indicating the number of restarts on the optimizer.
- **initializer:** list **of** float **or 'even'** – Initializes the starting targets for each component function. If 'even', every target is equal to y_train / len(use_columns).
- **report_down:** bool – If True, returns a DataFrame containing the rmse of training predictions vs actuals for each cycle.

Returns **self,** pandas DataFrame – Returns only the trained instance of self if report_rmse=False, otherwise returns the trained instance of self and a DataFrame containing the rmse values.

RSHDMRGPR.**predict**(*test_data, return_std=False*)

Predicts the results using the trained model parameters.

Parameters

- **test_data:** pandas DataFrame – The DataFrame containing the features of the test set. Should have one less column than the data used for training.
- **return_std:** bool – If True, the sum of all the component functions standard-deviation of the predictive distribution at the query points is returned along with the mean.

Returns

- **y_pred: 1D-**numpy Array – Sum of the mean of predictive distribution at query points of all component functions.
- **y_std: 1D-**numpy Array – Sum of the standard deviation of predictive distribution at query points of all component functions. Only returned when return_std=True.

RSHDMRGPR.**get_models**()

Returns the trained HDMR component functions.

Parameters

Returns list **of** GaussianProcessRegressor – List of GaussianProcessRegressor from the sklearn.gaussian_process library, their trained instances.

2.3 FirstOrderHDMRImpute class

class rshdmrgpr.**FirstOrderHDMRImpute**(*models, division=1000*)

Bases: object

This class is initialized by vectorizing the 1D HDMR component functions and creating a dictionary of output values with division subdivisions.

Parameters

- **models:** list of GaussianProcessRegressor – list of trained GaussianProcessRegressor models which represents the hdmr component functions of first order. Must be of first order (each has 1 input and 1 output).
- **division:** int – The number of divisions in the lookup table

2.3.1 FirstOrderHDMRImpute class methods

FirstOrderHDMRImpute.**get_table**()

Returns the lookup table for the component functions.

Parameters

Returns **panda** DataFrame – The lookup table for the HDMR component functions.

FirstOrderHDMRImpute.**get_yi**(*df_na*)

Modifies the DataFrame to contain the outputs of the first-order hdmr component functions. The output columns are concatenated past the last column of df_na.

Parameters

- **df_na:** pandas DataFrame – The DataFrame to impute. Must contain the output column and that column must be the last column.

Returns list – list of column indices for each row with missing df_na each corresponding to missing value column.

FirstOrderHDMRImpute.**impute**(*df_na, get_candidates=False, threshold=0.001*)

This function imputes the missing values given the input. Every single input row is expected to have at most one missing value.

Parameters

- **df_na:** pandas DataFrame – The DataFrame to impute. It is expected to contain the columns corresponding to 1D hdmr outputs (i.e. as an output of the get_yi function).
- **get_candidates:** bool – If True, returns all the candidates for imputation.
- **threshold:** float – The threshold distance to set for selecting candidates from look-up table. Correspond to δ in the paper (Section 2.2).

Returns

1) pandas DataFrame – The imputed DataFrame df_na if get_candidates=False.
2) pandas DataFrame, pandas Index, pandas Series, list of float – The imputed DataFrame df_na, the index of rows with null entries, the column names (indexed by the index of null entries) containing the null entry, and a list of candidates for imputing that missing value if get_candidates=True.

3 rshdmrgpr Examples

The supplementary IPython jupyter notebook (examples.ipynb) for this manual demonstrates in detail how these functions are used on several examples. All of the examples in the paper are worked out in this attached notebook.