User Guide for CSE700 Final Project: Kernel Ridge Regression

Uriel Garcilazo Cruz and Asma Jamali

April 8, 2025

Contents

1 Synopsis

Our project focuses on the implementation of global and local matrix representations of molecular properties (HOMO-LUMO gap and heat capacity) taken from the QM9 dataset. We use them in a comparative framework to evaluate the effects of trimming eigenvalues during the implementation of kernel ridge regression (KRR). Along with our main results, presented in the project: **Characterizing Overfitting in the Molecular Kernel-Based Methods**, we have developed a Python package to go along with this implementation.

Our software **krr** is an implementation of Kernel Ridge Regression (KRR) with support for custom kernels (Tanimoto, Dice, Gaussian and Laplacian) to predict molecular properties (e.g., HOMO-LUMO gap and heat capacity) from the QM9 dataset. The code integrates eigenvalue truncation to optimize performance.

2 How to run Python package

A functional folder for our application — with all the elements required to run the code — should have the following arrangement:

```
.
|-- CSE700
|   |-- Dataset
|   |   |-- bob_rep.npy
|   |   |-- coulomb_matrix_rep.npy
|   |   `-- dataset.pkl
|   |-- krr
|   |   |-- Kernel_ridge_regression.py
|   |   |-- plot_utils.py
|   |   `-- kernels_library.py
|   |-- main.py
|   `-- setup.py
```

Due to the size of the files contained in the folder Dataset, which is large relative to the conventional file size accepted by GitHub, the process of building the folder structure shown above requires two steps:

2.1 Clone the GitHub repository

In your terminal, navigate to the directory where you want to clone the repository and run:

```
git clone https://github.com/Asma-Jamali/CSE700.git
```

If you encounter any issues, please use your browser to navigate to the GitHub repository containing our KRR algorithm:

https://github.com/Asma-Jamali/CSE700/tree/KRR

In there, you will find the option to clone the repository. You can do this by clicking on the green button labeled "Code" and copying the URL provided. Keep in mind that you will need to have Git installed on your computer to clone the repository. If you don't have Git installed, you can download the repository as a ZIP file by clicking on the "Download ZIP" option in the same menu. This is especially useful for Windows users who may not have Git installed.

2.2 Download the dataset

The dataset is too large to be uploaded to GitHub. To download the dataset, please use the link below:

LINK TO DATASET

The dataset is password protected. Due to one of our developer's expertise being arachnology, the password is **Maratus_volans**.

Once the dataset has been downloaded, place it inside the folder and at the same level as main.py (control module).

2.3 Installing the package

The program is now in our local directory, but it's still not ready to run. There are multiple libraries that need to be installed. Luckily, Python has a straightforward way to ensure the interpreter can install such dependencies. Navigate to the folder where setup.py is located in the terminal and run:

```
pip install -e .
```

This will create an editable installation of the package, allowing you to modify the code and see the changes immediately without needing to reinstall the package. To install the package, you need to have Python version ¿=3.9, and pip installed on your system. If you don't have them installed, you can download Python from the official website: Python Downloads. Pip is included with Python installations. We also recommend using a virtual environment to avoid conflicts with other packages.

To execute the code, navigate to the folder where main.py is located in the terminal and run:

```
python main.py
```

If all the dependencies are installed correctly, the program should run without any issues.

2.4 Inputs and Outputs

The program takes the following inputs:

For the current version of the code, the user only need to specify the location of the picked dataset file, and the location for the output files:

```
# in main.py:
dataset_path = 'Dataset/dataset.pkl'
output_path = '/results/'
```

This data is a type of numpy array; a highly efficient data structure for storing large amounts of data.

The second most important set of parameters to feed into the kernel_ridge_regression function is the type of kernel and type of representation. A summary of potential valid inputs is presented in the next table:

Kernel	**Representation**	**Valid**	**Exception**
gaussian	CM	Y	
gaussian	BOB	Y	
laplacian	CM	Y	
laplacian	BOB	Y	
tanimoto	CM	N	Kernel type TANIMOTO of type CM is not supported.
tanimoto	BOB	N	Kernel type TANIMOTO of type BOB is not supported.
dice	CM	N	Kernel type DICE of type CM is not supported.
dice	BOB	N	Kernel type DICE of type BOB is not supported.

Table 1: Kernel Representations and Validity

If the user wants to opt for the use of the Tanimoto or Dice kernel, they MUST leave a missing representation in the call to the function **kernel_ridge_regrestion**.

```
kernel_ridge_regression(data, save_path="./results",
kernel_name='gaussian', mol_property='Gap', regularization=
False)
```

The example of the function above shows most of the default values for the parameters but representation is missing.

The program will then load the dataset and use it to train the KRR model. The model will be trained using the specified kernel (Tanimoto, Dice, Gaussian, or Laplacian) and the specified representation (if valid). The user will find two csv files:

- **results.csv**: This file contains the results of the truncated KRR model, including the R^2 scores, representing the performance of the model on both training and test samples.
- **eigenvalues.csv**: This file contains the eigenvalues of the kernel matrix.

The generated files will be deposited in the directory chosen by the user, with the type of analysis as a folder. For example, for a Gaussian regression type, the output folder will look as follows:

```
results/
```

```
|-- Gaussian
|    |-- eigenvalues.csv
|    `-- results.csv
```

variable **output_path**.

2.5 Plotting the results

The diagrams shown in the main manuscript for the project can now be generated by feeding the results.csv and eigenvalues.csv files into a plotting software. This can be achieved by feeding the data into a spreadsheet program like Excel or Google Sheets, or by using a custom script to generate the plots. We have made available a helper module that uses the package matplotlib to assist the user interested in using scripting for plotting the results.

To use it, the main function in the module — called plot_kernels.py — should receive one of two possible string inputs, determined based on the type of kernel used in the analysis:

Kernel Type	**Description**
fingerprint_based	If you have results.csv files for **tanimoto** and **gaussian** kernel types
distance_based	If you have results.csv files for **gaussian** and **laplacian** kernel types

Table 2: Inputs for helper module `plot_utils.py`

Please keep in consideration that the helper module runs with the assumption that your you have the result files for EACH of the two kernels (e.g. gaussian and laplacian). This is due to the comparative nature of the plotting algorithm, designed to highlight the differences between the two comparable kernel types.

For example, if your results folder structure looks as follows:

```
results/
|-- gaussian
|    |-- eigenvalues.csv
|    `-- results.csv
|-- laplacian
|    |-- eigenvalues.csv
|    `-- results.csv
```

Then you would call the plotting function with the following parameters:

```
plot_kernels(kernel_type="distance_based", path_results="..path
/results/", regularization=False)
```

For the purposes of demonstration, we have incorporated a call to this method in the main.py file. The end user can choose to comment it out or leave it as is.

In conclusion, this script will generate the plots and save them in a folder called Figures. For examples on how to analyze and interpret the results, please refer to the main project's manuscript: **Characterizing Overfitting in the Molecular Kernel-Based Methods**