Recursive Maxima Hunting in Feature Selection

1 Introduction

Dimensionality reduction is a critical aspect of developing effective machine learning methods for automatic induction, particularly when dealing with functional data. The need to reduce information while preserving the underlying structure of the data is essential for improving model interpretability, computational efficiency, and predictive accuracy. Traditional dimensionality reduction techniques in functional data analysis (FDA), such as principal component analysis (PCA) [1] and partial least squares (PLS) [2], are often employed to address these issues. However, variable selection methods have emerged as an alternative approach to simplifying complex functional data.

Variable selection techniques aim to replace the complete function X(t) by a lower-dimensional vector (X(t1), ..., X(td)) for a set of suitably chosen points {t1, ..., td}, where d is small. Although a majority of the existing literature on feature selection in supervised learning with functional data has focused on regression problems, recent works have explored classification problems as well. These studies have investigated various feature selection methods, including lasso, mutual information-based approaches, and wrapper techniques, among others.

In this project report, we will discuss the recursive maxima hunting (RMH)[3] method, a novel variable selection technique for functional data classification. RMH is a recursive extension of maxima hunting (MH)[4], which selects variables based on the maxima of a relevance function that measures the correlation between the predictor functional variable and the class label. By iteratively removing information associated with the selected variables, RMH has demonstrated comparable or superior predictive accuracy to standard dimensionality reduction techniques like PCA and PLS, as well as other state-of-the-art feature selection methods for functional data.

We will explore the key contributions and advancements of RMH in the context of existing works on feature selection for functional data, examining its benefits, drawbacks. We will write our own program codes for MH and RMH and evaluate their performance in various Brownian simulation data as well as some well known dataset for functional data. Furthermore, we will analyze the method's performance in comparison with other variable selection approaches such as PCA and PLS, highlighting the advantages of adopting a fully functional perspective when dealing with continuous functional data. Through this project, we have implemented Maximum Hunting algorithms and Recursive Maximum Hunting algorithms. We have uploaded all the code into Github. Through the implementation and comparison between different dimension reduction algorithms, we gain a comprehensive understanding of the current state of research on dimensionality reduction and recursive maxima hunting for functional data classification.

2 Method

2.1 Maxima Hunting (MH)

Maxima Hunting is a filter variable selection method for problems with a target variable. It evaluates a dependence measure between each point of the function and the target variable, and keeps those points in which this dependence is a local maximum.
Different measures of dependency can be used for this purpose. In Berrendero et al. (2016b), the authors propose the distance covariance between the random variables $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$ is

$$\mathcal{V}^2(X,Y) = \int_{\mathbb{R}^{p+q}} |\varphi_{X,Y}(u,v) - \varphi_X(u)\varphi_Y(v)|^2 w(u,v)dudv$$

where $\varphi_{X,Y}$, φ_X , φ_Y are the characteristic functions of (X,Y), X and Y , respectively, $w(u,v) = (c_p c_q |u|_p^{1+p} |v|_q^{1+q})^{-1}$, $c_d = \frac{\pi^{(1+d)/2}}{\Gamma((1+d)/2)}$ is half the surface area of the unit sphere in $\mathbb{R}^{d+1}$, $|\cdot|_d$ stands for the Euclidean norm in $\mathbb{R}^d$.The square of the distance correlation is

$$\mathcal{R}^2(X,Y) = \begin{array}{ll} \frac{\mathcal{V}^2(X,Y)}{\sqrt{\mathcal{V}^2(X,X)\mathcal{V}^2(Y,Y)}}, & \mathcal{V}^2(X)\mathcal{V}^2(Y) > 0 \\ 0, & \mathcal{V}^2(X)\mathcal{V}^2(Y) = 0. \end{array}$$

The specific implementation steps are as follows. Firstly, smooth each curve in the dataset using appropriate smoothing techniques to reduce noise and capture the underlying structure of the functional data. Then, identify local maxima for each smoothed curve by using $\mathcal{R}^2(X,Y)$ scores. For each local maximum in the mean curves, compute the difference between the maximum value and the corresponding value in the other class's mean curve. This difference serves as a score to evaluate the importance of each maximum in differentiating between classes. Finally, rank the local maxima according to their $\mathcal{R}^2(X,Y)$ scores, and select a subset of the most significant maxima. These selected maxima are then used as features in the classification task.

$$X(t_i) = \underset{t\in[0,1]}{\operatorname{argmax}} \mathcal{R}^2(X(t),Y), \quad i = 1,2,\ldots,d.$$

Selecting the local maxima serves two purposes. First, it ensures that the points that are relevant in isolation are selected, as they must maximize their dependence with the target variable. Second, the points that are relevant only because they are near a relevant point (and are thus highly correlated with it) are NOT selected, as only local maxima are selected, minimizing the redundancy of the selected variables.

2.2 Recursive Maxima Hunting（RMH）

Recursive maxima hunting is an extension of the maxima hunting method for variable selection in functional data classification. The maxima hunting algorithm introduced in section 2.1 partially avoids redundancies, but the local approach cannot distinguish redundancies between different local maxima. Furthermore, there could be examples that are highly relevant but not the maxima of the relevance functions. The primary goal of this technique is to select the maxima of relevance function $R^2(X(t), Y)$ in a recursive way while excluding the redundant information that is associated with the selected maxima which is done by subtracting the conditional expectation $E(X(t)|X(t0))$ from $X(t)$. We assume that this process is Brownian with the following conditional expectations:

$$\mathbb{E}(X(t)|X(t_0)) = \frac{\min(t, t_0) - t\,t_0}{t_0(1-t_0)} X(t_0) = \begin{cases} \frac{t}{t_0} X(t_0), & t < t_0 \\ \frac{1-t}{1-t_0} X(t_0), & t > t_0. \end{cases}$$

With this technique RMH further refined the variable selection process and improved classification performance by incorporating a recursive search approach.

The detailed steps of recursive maxima hunting technique and its corresponding pseudo code is shown below in Algorithm 1. There are two hyperparameters in this algorithm. The stopping threshold s will ignore all the irrelevant values that have relevance scores lower than s which is typically set to s = 0.95. The redundancy threshold r will exclude all the points that are relevant to the maxima, in practical, r is set to 0.05.

For the first step of RMH, we start with the maxima hunting procedure as explained in section 2.1 to find the point with maximum relevance values. Then we compare the maxima value with threshold s, if it’s higher than the threshold, we keep it as a selected feature. Otherwise, we will stop the algorithm since the rest points are irrelevant. This corresponds to line 8 to 13 in the pseudo code. Then after saving the selected maxima point, the algorithm split the function data into 2 parts, the left part with t less than the maxima time and the right part with t larger than the maxima time. The redundancy exclusion technique is then applied on the two sets. The relevance values between $X(t)$ and $X(t\ max)$ are calculated and only the points with relevance lower than threshold r will remain. Afterwards, we call the RMH algorithm on these two sets to recursively select more maxima points. The pseudo code related to this step is shown from line 14 to 25. Eventually, we will use all the selected maxima points as the reduced features for KNN to classify the trajectory.

Algorithm 1 Recursive Maxima Hunting

```
 1: function RMH($X(t), Y$)
 2:     $\mathbf{t}^* \leftarrow [\,]$                                      ▷ Vector of selected points initially empty
 3:     RMH_rec($X(t)$,$Y$,0,1)                          ▷ Recursive search of the maxima of $\mathcal{R}^2(X(t), Y)$
 4:     return $\mathbf{t}^*$                                      ▷ Vector of selected points
 5: end function
 6: procedure RMH_REC($X(t), Y, t_{inf}, t_{sup}$)
 7:     $t_{max} \leftarrow \underset{t_{inf} \le t \le t_{sup}}{\operatorname{argmax}} \left\{\mathcal{R}^2(X(t), Y)\right\}$
 8:     if $\mathcal{R}^2(X(t_{max}), Y) > s$ then
 9:         $\mathbf{t}^* \leftarrow [\mathbf{t}^*\ t_{max}]$               ▷ Include $t_{max}$ in $\mathbf{t}^*$ the vector of selected points
10:         $X(t) \leftarrow X(t) - \mathbb{E}(X(t) \mid X(t_{max})),\ t \in [t_{inf}, t_{sup}]$
11:     else
12:         return
13:     end if
14:     ▷ Exclude redundant points to the left of $t_{max}$
15:     $t^-_{max} \leftarrow \max_{t_{inf} \le t < t_{max}} \left\{t : \mathcal{R}^2\left(X(t_{max}), X(t)\right) \le r\right\}$
16:     if $t^-_{max} > t_{inf}$ then
17:         RMH_rec($X(t), Y, t_{inf}, t^-_{max}$)                    ▷ Recursion on left subinterval
18:     end if
19:     ▷ Exclude redundant points to the right of $t_{max}$
20:     $t^+_{max} \leftarrow \min_{t_{max} < t \le t_{sup}} \left\{t : \mathcal{R}^2\left(X(t_{max}), X(t)\right) \le r\right\}$
21:     if $t^+_{max} < t_{sup}$ then
22:         RMH_rec($X(t), Y, t^+_{max}, t_{sup}$)                    ▷ Recursion on right subinterval
23:     end if
24:     return
25: end procedure
```

Recursive maxima hunting is designed to find an optimal subset of maxima that yields the best classification performance, taking into account the specific classifier used and the potential interactions between maxima. This method provides a more refined variable selection process compared to the basic maxima hunting technique, leading to improved classification results and a better understanding of the most important features in the functional data.

2.3 Principal Component Analysis (PCA)

PCA is an unsupervised linear dimensionality reduction technique that aims to identify the most significant directions in the feature space, where the data exhibits the maximum variance. It is widely used for data compression, visualization, and noise reduction. The PCA algorithm involves the following steps:

Firstly, scale the features to have zero mean and unit variance and compute the covariance matrix of the standardized data. Then perform eigenvalue decomposition on the covariance matrix, which yields eigenvectors (principal components) and eigenvalues (explained variance). Secondly, sort the eigenvectors in descending order of their corresponding eigenvalues and select the top k eigenvectors with the highest eigenvalues, where k is the desired reduced dimension. Finally, project the original data onto the selected eigenvectors to obtain the lower-dimensional representation.

PCA is a powerful technique for identifying the underlying structure in the data, but it does not consider the relationship between the input features and the response variable (class label or target variable) during the dimensionality reduction process.

2.4 Partial Least Squares (PLS)

PLS is a supervised linear dimensionality reduction method that seeks to find the directions in the feature space that not only explain the maximum variance in the input data but also have the strongest correlation with the response variable. PLS is often used in regression and classification tasks, especially when there is a high degree of multicollinearity among input features.

For the implementation of the PSL, firstly, standardize the input dataset and response and compute the covariance matrix between them. Then, perform Singular Value Decomposition (SVD) on the covariance matrix, yielding the loading vectors for both the input data and the response variable, and compute the PLS components by projecting the input data and the response variable onto their respective loading vectors. Finally, determine the optimal number of PLS components using cross-validation or other model selection techniques, and use these components as the lower-dimensional representation for further analysis, such as regression or classification.

3 Experiments

To assess the performance of Recursive maxima hunting, we have carried out experiments both in simulation data and real-world data. We compared our RMH algorithm with maxima hunting, PCA and PLS algorithm through all the scenarios. In all these experiments, after applying the dimension reduction algorithms, the K-nearest neighbor with L2 distance is utilized for classification. The K value is tuned by 10-fold cross validation selecting from the set [1, $\sqrt{N}$] where N is the size of training data.

3.1 Simulation data

For the simulation data, we generate data from the Brownian motion B(t) with different deterministic trends m(t).

$$\begin{cases} P_0 : B(t) & , \quad t \in [0,1] \\ P_1 : B(t) + m(t) & , \quad t \in [0,1] \end{cases},$$

The standard Brownian motion P0 is shown in Fig 1 which is labeled as 0. The deterministic trajectory is labeled as 1 and we draw 50% samples from the standard Bronian motion and the other 50% samples from the deterministic trajectory. Our goal is to discriminate the trajectories using dimension reduction algorithms and KNN classifiers. Peak 1 in Fig 1 is an example from maxima hunting algorithm [4]. The deterministic trend is defined as $m(t) = 2\phi_{3,3}(t)$ where

$$\Phi_{m,k}(t) = \int_0^t \sqrt{2^{m-1}} \left[\mathbb{I}_{\left(\frac{2k-2}{2^m}, \frac{2k-1}{2^m}\right)}(s) - \mathbb{I}_{\left(\frac{2k-1}{2^m}, \frac{2k}{2^m}\right)}(s) \right] ds, \quad m, k \in \mathbb{N}, 1 \le k \le 2^{m-1}.$$

The black line in the figure is the average of all the trajectories. For Peak 2, we use the trend $m(t) = 2\phi_{3,2}(t) + 3\phi_{3,3}(t) - 2\phi_{2,2}(t)$ which is also illustrated in Fig 1. The rest four trajectories are based on the following deterministic functions: Square: $m(t) = 2t^2$; Sin : $m(t) = 1/2\, sin(2\pi t)$; Tanh : $m(t) = 4 * tanh(t)$; exp: $m(t) = 1 - exp(t)$. The trajectory distributions are illustrated in Figure 2.

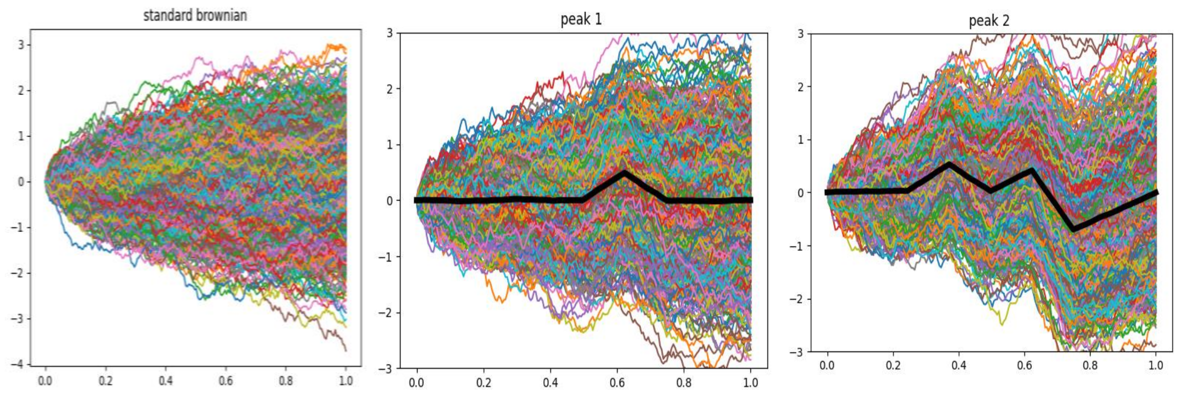

Figure 1: Trajectories for Standard Brownian , Peak 1 and Peak 2.

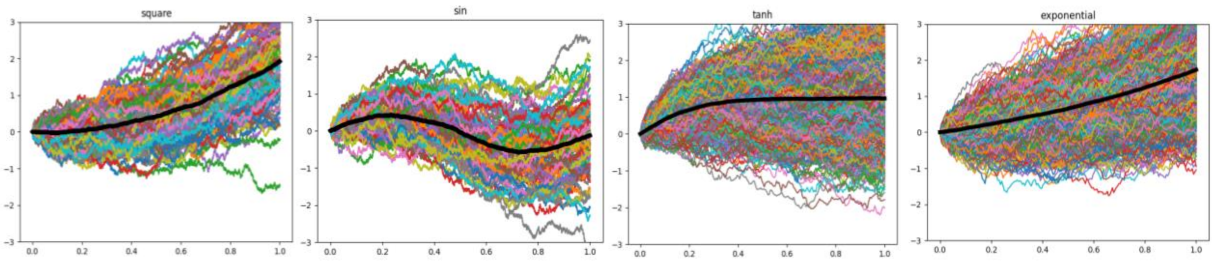

Figure 2: Trajectories for Square, Sin , Tanh and Exponential

Figure 3 displays the average classification error for different algorithms under different training data size N = {50,100,200,500,1000}. The Base represents the experiment that directly applies KNN on the raw data. The test set’s size is fixed to 1000 and is generated independently. The classification errors reported in Figure 3 are the average errors among 200 independent repetitions. The first four experiments are the same from the RMH paper [3] which are peak1, peak2,Sin and Square. Beyond the experiments from the paper, we also generated two extra examples for tanh and exponential curves. Figure 4 displays the averaged dimension or selected components after applying each algorithm. The lower the dimension, the better the computational and storage efficiency we will have.

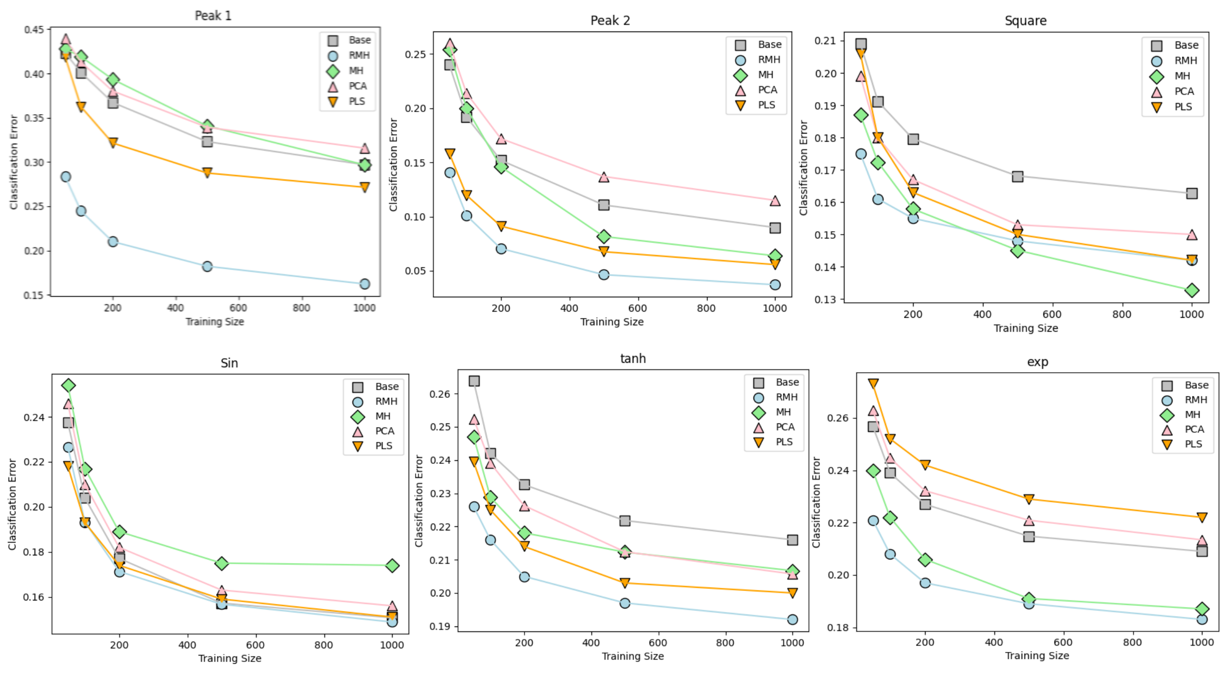

Figure 3: Average Classification error under different training sizes.

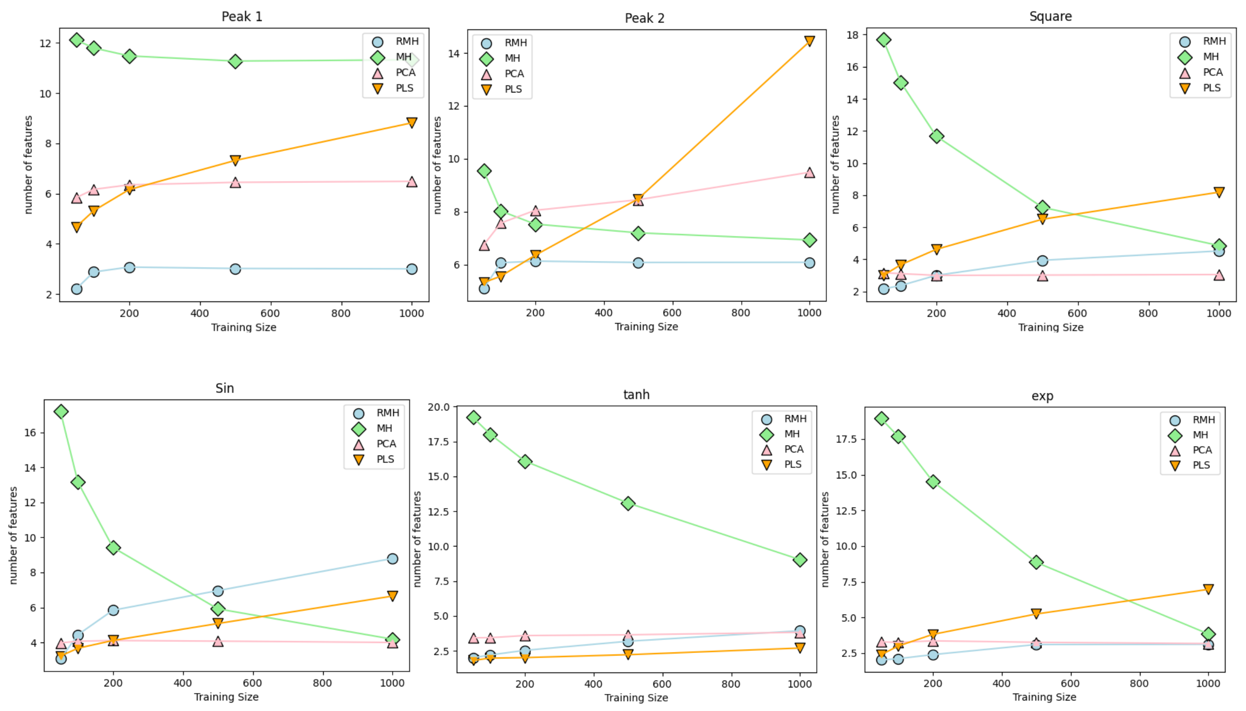

Figure 4: Average reduced dimension (selected variables) under different training sizes

From the results reported from Figure 3 and Figure 4, we can conclude that RMH has the best overall performance. First, the RMH's error rate is always lower than the Base which means applying RMH will not reduce informative components and lower the performance of the classifier especially in the peak examples. The RMH algorithm outperforms the traditional PCA and PLS method since it is specifically designed for functional data and better fits to trends that have maximums. The RMH also mitigates the weakness of MH which its previous algorithm that might focus on the maximum too much. The performance is downgraded in periodical functions that have both valleys and peaks such as sine function. Other than the experiment mentioned in RMH [3] , we performed two extra simulations on tanh and exponential function which further validates the results. Another advantage of RMH is the size of the selected component. Due to the recursive exclusion of redundant points near the maxima, RMH ignores many features that are close to the maxima which would be selected by the MH algorithm. Thus through all the examples, RMH can always achieve a lower number of variables than almost all the algorithms. Overall, our simulation experiments indicate that RMH can beat most other algorithms both in accuracy and dimension size.

3.2 Real-World data

To assess the performance of RMH in real-world functional classification problems, we have carried out a second batch of experiments in four datasets.

The Berkeley Growth Study (Tuddenham and Snyder, 1954) recorded the heights of 54 girls and 39 boys between the ages of 1 and 18 years. Each sample has 31 features, which are heights collected from different ages, and the standard error of these measurements was about 3mm. The label is gender.

The *Tecator* dataset consists of 240 finely chopped meat samples. Each sample data consists of the absorbance spectra of 100 points selected equidistantly from 850 to 1050 mm in wavelength. The label is the fat content. We label meat with a fat content of more than 20% as fat meat, and vice versa as lean meat. Shown as figure 5, we found that the curves are fairly smooth and the shape of the curve may contain more important information.

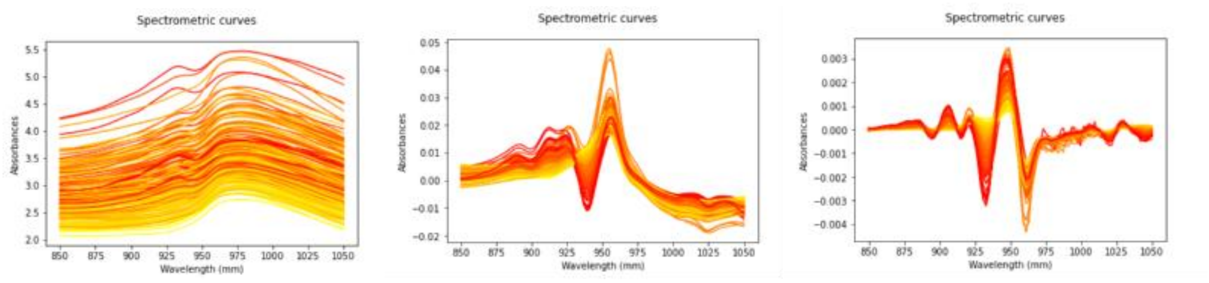

Figure 5: original data, first derivative data, second derivative data of *tecator*

After experiments, we found that the KNN score of the second order derivative data is higher:

	original data	**first derivative data**	**second derivative data**
KNN bandwidth	3	4	2
KNN score	0.3501	0.9428	0.9469

So we use the second order derivative for classification.

The *Phoneme* dataset was formed by selecting five phonemes for classification based on digitized speech from this database. It has 4509 speech frame samples. Each speech frame is represented by 512 samples at a 16kHz sampling rate, and each frame represents one of the above five phonemes. Here, we consider the binary problem of distinguishing between the phonemes “aa” and “ao”. The phoneme curves are very irregular and noisy, so we usually will want to smooth them as a preprocessing step. We chose the Nadaraya-Watson smoothing method. The bandwidth parameter controls the influence of more distant points on the final estimation. So, it is to be expected that with larger bandwidth values, the resulting function will be smoother. Figure 6 are examples of under smoothing (with bandwidth = 0.05) and over smoothing (with bandwidth = 1.0) using the Nadaraya-Watson method with normal kernel.

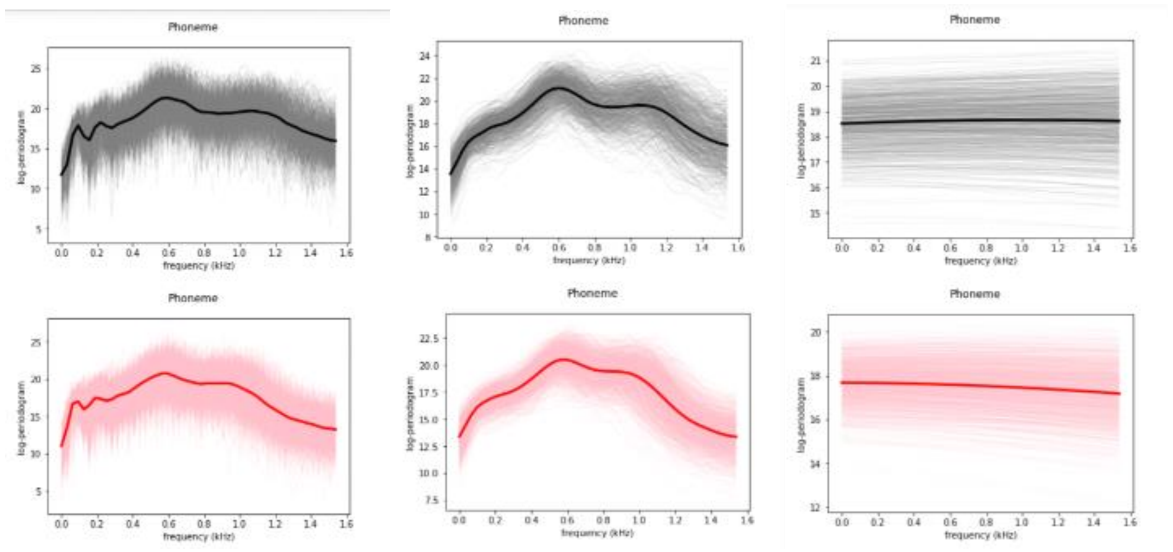

Figure 6:original data,under smoothing and over smoothing data of *phoneme*

After several experiments, we chose a bandwidth=0.2 that can make the KNN score better.

bandwidth	**0.1**	**0.2**	**0.3**	**1.0**
KNN score	0.7848	0.7907	0.7849	0.7790

The *Medflies* dataset recorded the daily egg-laying patterns of 512 flies. Each sample has 30 features, representing the number of eggs laid by flies per day during the 30-day period from the 5th day to the 34th day of birth. The label is used to distinguish between short-lived and long-lived flies.

Growth and *Tecator* are better classified datasets. Therefore, they are relatively easy problems. In contrast, *Phonemes* and *Medflies* are notoriously difficult classification tasks. Figure 7 plots some sample trajectories for different labels in each dataset and the corresponding mean values (thick lines).

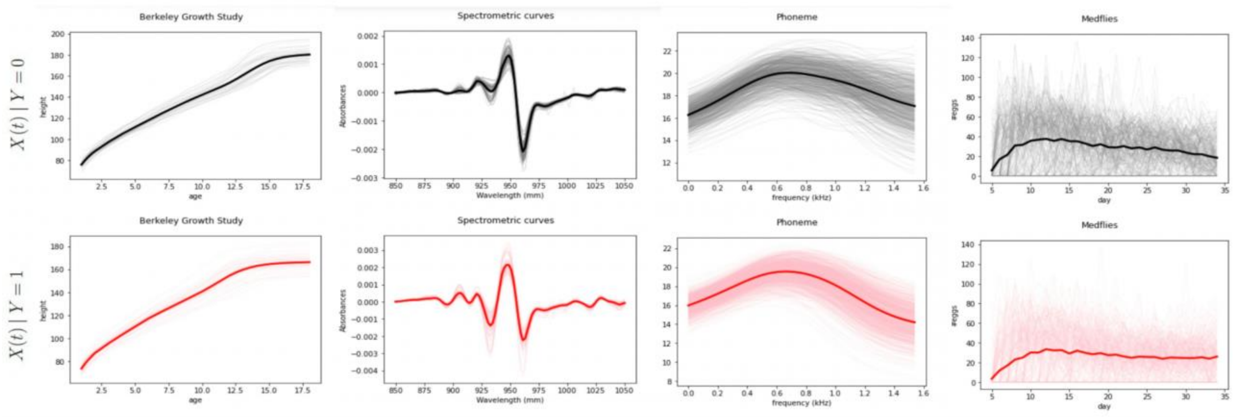

Figure 7:some sample trajectories for different labels in each dataset and the corresponding mean values

To estimate the classification error, the dataset is randomly split into a training set (with 2/3 of the observations) and a test set (1/3). The procedure is repeated 200 times. The resulting boxplots for each dataset and method are shown in Figure 8.

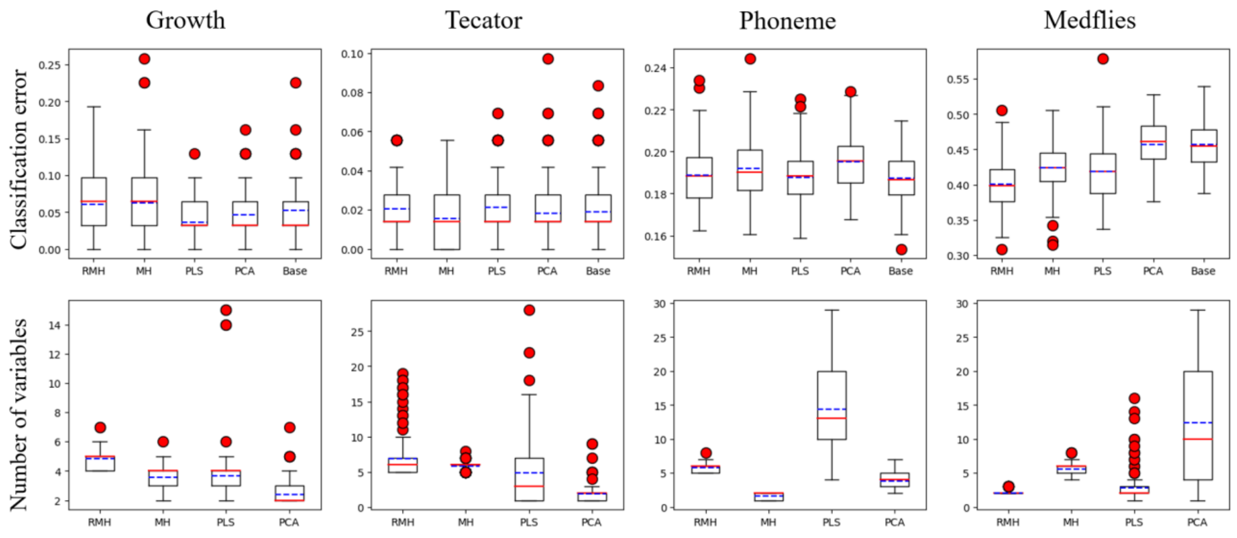

Figure 8 :Classification error (first row) and number of variables/components selected (second row)

From figure 8, we can notice that, in general, dimensionality reduction is effective: the accuracy of the four considered methods is similar or better than the *Base* method, in which complete trajectories are used for classification. For *Medflies* data, *Base* does not perform well as the trajectories are not smooth and PCA, PLS, MH has better performance. The best overall performance corresponds to RMH. In the easy problems (*Growth* and *Tecator*), all methods behave similarly and give good results. In *Growth*, RMH tends to select more variables than the other methods. In the more difficult problems (*Phoneme* and *Medflies*), RMH exhibits the best performance among four algorithms. In *Phoneme*, *Base* is more accurate compared to these four reduction algorithms. In *Medflies*, RMH yields very accurate predictions while selecting only two variables. Compared to MH, RMH performed much better in both *Phoneme* and *Medflies*. We can conclude that the accuracy of RMH is comparable and often better than state-of-the-art functional classification methods. In any case, RMH will give better performance if used in combination with other types of classifiers or adapted and used as a wrapper or, even, as an embedded variable selection method.

4 Discussion & Conclusion

In this section, we will talk about the challenges we meet and result discussion compared to the RMH original paper, the conclusion will be given at the end.

4.1 Key Challenges

Several challenges occurred during the progress of our RMH project. First will be the implementation of the recursive maxima hunting algorithm. It takes effort to understand the relevance function and transfer the conditional expectation formula into codes. Due to the recursive nature of this algorithm, it also takes extra time to debug through each recursive call.

Other than the implementation challenges, we have also faced challenges on training and fine tuning hyperparameters for different algorithms. In KNN classification, in order to find the best K value, we use GridSearchCV to tune hyperparameters. We set the value of k_neighbors range from the set [1, $\sqrt{N}$] where N is the size of training data. Given a different set of hyperparameters, GridSearchCV loops through all possible hyperparameter values and combinations and fits the model on the training dataset. In this process, it is able to determine the best values and combinations (from a given set of parameters) that result in the best precision.

In the PCA method, we tried different values of n_components and compared the classification error with paper. We noticed that if we choose the best values, the classification error would be much lower, but the dimension is quite high. By validation step, we finally set n_components as

0.92 in real data and 0.95 in simulation data, which gives a good balance between dimension and error result.

In MH, the smoothing parameter was selected in an approximately optimal way. Through many experiments, we found that the larger the smooth value, the more the dimension reduced, but the classification accuracy did not improve, or even decreased. So We carried out cross-validation on the training set to choose an optimal smooth parameter. In this way, we can get reasonable error data as well as dimension size.

In the PLS method, the number of PLS components, namely the number of features selected, is determined by using 70% data for training and 30% data for testing and comparing the R^2 score of prediction error for the number of PLS components change from 1 to 30. The best way of selecting the number of PLS components is to use cross-validation, but considering the computational burden, we just directly split the dataset into train and test data. However, it still gives a relatively good performance.

4.2 Discussion

When comparing Maxima Hunting to PCA and PLS, it's essential to consider the specific context and the type of data you are working with. Maxima Hunting is specifically designed for functional data and focuses on selecting discriminative features based on local maxima, while PCA and PLS are more general dimensionality reduction techniques that work with a wide range of data types.

Comparing our experiment result with the original paper, for the result of building model on the simulation data, we can see that the RMH method has the overall smallest classification error considering all the four datasets, and the value of the error is similar to that of the original paper; meanwhile, the performance of PCA and PLS is better than that displayed on the paper, and PLS outperform BASE on the Peak, Peak2, and Square dataset in our experiment. As for the performance of dimension reduction, RMH has relatively good performance. However, its performance on the sin dataset could not be realized in our experiment.

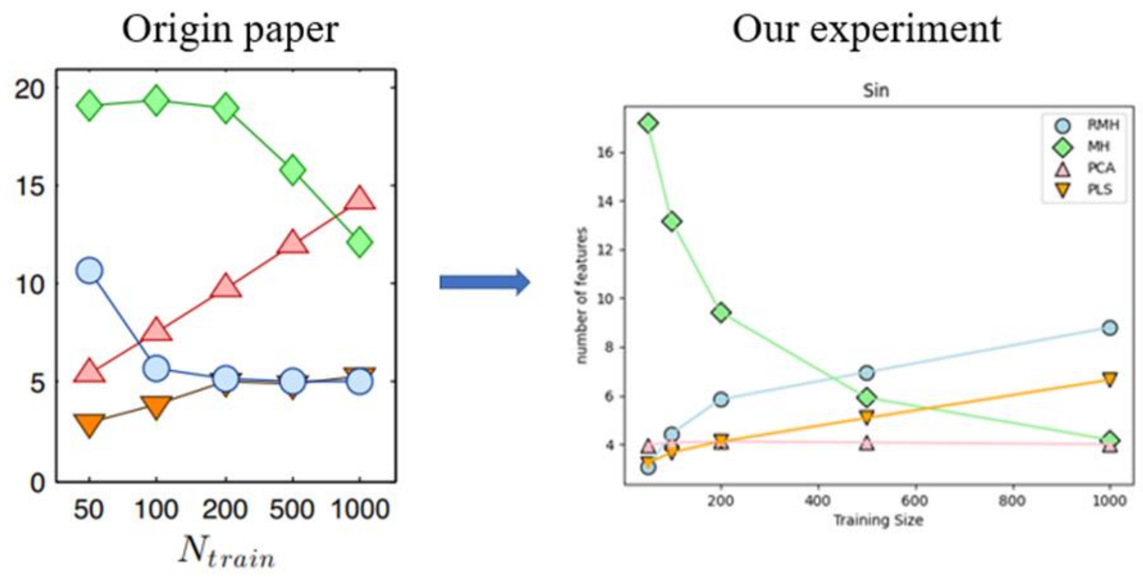

Figure 9 comparison between result of original paper and our experiment on *sin* dataset

For the real dataset, the classification error gotten in our experiment is very similar to that in the original paper. The effect of feature selection stated in the original paper is also able to be implemented in our experiment, except on the dataset Phoneme, where we can see that mostly the number of variables RMH reduces to is 6, but not 2.

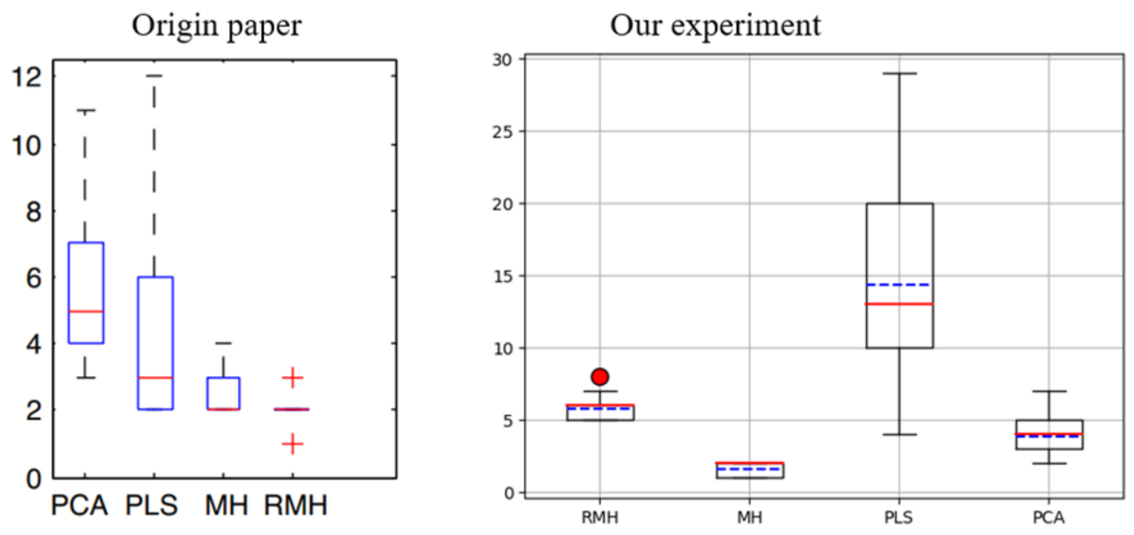

Figure 10 comparison between original paper and our results on the *Phoneme* dataset

4.3 Conclusion

To conclude, Recursive Maxima Hunting is a method for variable selection in functional data classification that avoids redundancies in the dataset by recursively selecting a local maxima and eliminating the influence of the selected point in the next recursion each time. It is useful in dealing with real-world data like functional magnetic resonance imaging, near-infrared spectra, and so on. The paper performed experiments for RMH on both simulated and real-world datasets, compared its effect with Maxima Hunting, Partial Least Squares, and Principal Component Analysis, and used KNN for classification. The paper got the result that RMH has a better performance than other three methods most of the time. We tried to realize the experiment by ourselves and found that most of the conclusions obtained by the paper are true, however, it has a worse performance on the sin and Phoneme dataset than it displayed on the paper. Besides, we carried out extra experiments on two simulated datasets named exp and tanh and found that RMH also has a relatively good performance compared with other methods, especially when the sample size is small. Overall, RMH is a dimensional reduction method that can handle the functional data, and have a good performance compared with some traditional dimensional reduction algorithms.

Reference

[1] Ramsay, J. O. and B. W. Silverman (2005). Functional Data Analysis. Springer.
[2] Preda, C., G. Saporta, and C. L´ev´eder (2007). PLS classification of functional data. Computational Statistics 22(2), 223–235.
[3] Jos´e L. Torrecilla and Alberto Su´arez. Feature selection in functional data classification with recursive maxima hunting
[4] Berrendero, J. R., A. Cuevas, and J. L. Torrecilla (2016b). Variable selection in functional data classification: a maxima hunting proposal. Statistica Sinica 26(2), 619–638.