Received: 20 July 2023 | Revised: 21 September 2023 | Accepted: 10 November 2023 | Published online: 10 November 2023

RESEARCH ARTICLE

Artificial Intelligence and Applications
2024, Vol. 2(1) 38–44
DOI: 10.47852/bonviewAIA32021381

BON VIEW PUBLISHING

# KELL: A Kernel-Embedded Local Learning for Data-Intensive Modeling

**Changtong Luo**[1,*]

[1]*Institute of Mechanics, Chinese Academy of Sciences, China*

**Abstract:** Kernel methods are widely used in machine learning. They introduce a nonlinear transformation to achieve a linearization effect: using linear methods to solve nonlinear problems. However, typical kernel methods like Gaussian process regression (GPR) suffer from a memory consumption issue for data-intensive modeling: the memory required by the algorithms increases rapidly with the growth of data, limiting their applicability. Localized methods can split the training data into batches and largely reduce the amount of data used each time, thus effectively alleviating the memory pressure. This paper combines the two approaches by embedding kernel functions into local learning methods and optimizing algorithm parameters including the local factors and model orders. This results in the kernel-embedded local learning (KELL) method. Numerical studies show that compared with kernel methods like GPR, KELL can significantly reduce memory requirements for complex nonlinear models. And compared with other non-kernel methods, KELL demonstrates higher prediction accuracy.

**Keywords:** kernel methods, Gaussian process regression, data-intensive modeling, local learning, KELL, complex nonlinear models, prediction accuracy

## 1. Introduction

Kernel methods have been widely applied in the field of machine learning (Hofmann et al., 2008; Schölkopf & Smola, 2018). The main idea is to introduce a nonlinear transformation to achieve a linearization effect, so as to solve nonlinear problems using linear methods. These methods have achieved remarkable results in many practical applications, such as support vector machines (SVMs) (Cortes & Vapnik, 1995; Vapnik, 2013) and Gaussian process regression (GPR) (Rasmussen & Williams, 2005). However, typical kernel methods like GPR face a memory usage problem for data-intensive modeling: as the amount of training data increases, the memory resources consumed by the algorithm increase rapidly, which limits the application of the algorithm in data-intensive modeling scenarios (Lawrence, 2009; Quiñonero-Candela & Rasmussen, 2005).

On the other hand, localization methods are also an effective learning strategy. The basic idea is to divide the training data into batches, which greatly reduces the amount of data used each time, thereby effectively alleviating memory pressure. Localization methods have been successfully applied in many fields, such as online learning (Cesa-Bianchi & Lugosi, 2006; Shalev-Shwartz, 2012) and incremental learning (Muhlbaier et al., 2009; Polikar, 2006). However, these methods are often limited in dealing with high-dimensional complex nonlinear problems, because they do not use kernel functions to achieve linearization (Bousquet & Elisseeff, 2002; Schölkopf et al., 1999).

More recent related works are mainly focused on improving the scalability of existing methods. Explicit feature maps have emerged as a substitute for traditional kernel-based methods (Francis & Raimond, 2021). The neural network approximation captures the location, orientation, and shape of the solution transition near a localization, while the standard RK approximation models the smooth part of the solution (Baek et al. 2022). Matrix-induced regularization is suggested to enhance the complementarity between base kernels (Qiu et al. 2023).

This paper proposes a new method that combines kernel functions with local learning methods, thus achieving kernel-embedded local learning (KELL). Specifically, we embed kernel functions in local learning methods and optimize algorithm parameters like local factors and model orders to improve model prediction accuracy. In other words, this work also focuses on improving the scalability so that large amounts of data could be properly handled in a reasonable time, without losing the quality of the results.

To verify the effectiveness of the KELL method, we first take a set of synthetic data as an example to demonstrate the superiority of the KELL method in dealing with complex nonlinear models. For this dataset, we find that compared with kernel methods like GPR, the KELL method can greatly reduce memory requirements; compared with other non-kernel methods, the KELL method has higher prediction accuracy.

Next, we apply the KELL method to real datasets and compare it with other kernel methods and non-kernel methods. The results show that while reducing memory requirements, the KELL method can still maintain high prediction accuracy. This indicates that the KELL method has broad application prospects.

## 2. KELL Method

### 2.1. Algorithm description

This paper proposes a multivariate local learning algorithm with embedded kernel functions (i.e., KELL). The goal is to build a model

*Corresponding author: Changtong Luo, Institute of Mechanics, Chinese Academy of Sciences, China. Email: luo@imech.ac.cn

that can accurately predict unknown data based on known datasets and pending datasets, with less memory cost. To achieve this goal, the following steps are taken:

First, perform some necessary data transformations on the known and pending datasets, such as standardization and normalization, to facilitate subsequent calculations and analysis. Second, divide the known dataset into three subsets: training set, validation set, and test set. The training set is used to construct the model, the validation set is used to evaluate the learning effect of the model, and the test set is used to calculate the model's prediction performance metrics. Then, calculate the distance between each point in the validation set and each point in the training set, and store them in a distance matrix. This distance matrix is an important basis for subsequent steps.

Next, determine the optimal neighborhood dataset, kernel function weights, and multivariate low-order model by adjusting two important parameters: local factor and model order. The local factor determines the size of the neighborhood dataset for each validation point, and the model order determines the complexity of the multivariate low-order model. The model order could be 0, 1, or 2, which correspond to moving average, linear, and quadratic models, respectively. The neighborhood dataset refers to a portion of the training points that are closest to the validation point. The number of neighboring points ($N_{neighbors}$) in the neighborhood dataset depends on a preset local factor range, which is [0.05, 0.5] in this paper. The kernel function weights ($K_\lambda(x_{new}, x_i)$) refer to the weights assigned to each neighboring point according to the kernel function and distance matrix. The multivariate low-order model ($f_\theta(\cdot)$) refers to a weighted multivariate polynomial function constructed using the neighborhood dataset and given model order.

The local estimate (regression) at a new location $x_{\text{new}}$ is defined as $\hat{y} = f_{\tilde{\theta}}(x_{\text{new}})$, where $\tilde{\theta}$ minimizes the residual sum-of-squares (RSS)

$$\text{RSS}(f_\theta, x_{new}) = \sum_{i=1}^{N_{neighbors}} K_\lambda(x_{new}, x_i)(y_i - f_\theta(x_i))^2$$

Note that only "local" kernel matrices are computed to make predictions at new locations. That is, only a small part data (not all data) are used to construct the local model. Thus, it requires much less memory.

To find the optimal parameter combination, the following steps need to be repeated under different parameter values: determine neighborhood dataset, determine kernel function weights, construct multivariate low-order model, and evaluate learning effect. The learning effect is evaluated by calculating the coefficient of determination $R^2$ of the model on the validation set. The larger this coefficient, the better the model can explain the variation in the validation data. When the parameter combination that maximizes $R^2$ is found, the tuning can be stopped. The evaluation of the learning effect is based on the test set. Calculate the model's prediction performance metrics, i.e., the coefficient of determination $R^2$ of the model on the test set. The coefficient of determination $R^2$ is determined by the RSS and total sum-of-squares (TSS) on the test set defined as $R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$, where $\text{RSS} = \sum_{i=1}^{N_{\text{test}}} (y_i - \hat{y}_i)^2$, $\text{TSS} = \sum_{i=1}^{N_{\text{test}}} (y_i - \bar{y})^2$, and $\bar{y} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} y_i$. The larger this coefficient ($R^2$), the better the model can predict the variation in unknown data.

Finally, fine-tune the model using all known data under the optimized local factor and model order and recalculate the model coefficients. Then, use the fine-tuned model to calculate the predicted results of the pending dataset, and inverse transform the predicted results back to the original data scale. At the same time, output the prediction performance metrics to evaluate the model.

## 2.2. Algorithm steps

The algorithm steps for multivariate local learning with embedded kernel functions (i.e., KELL) are as follows:

1) Input: Read known dataset, pending dataset
2) Data transformation: Perform standardization, normalization on known and pending datasets
3) Data splitting: Split known dataset into training set, validation set, and test set
4) Calculate distance matrix: Calculate distance between each point in validation set and training set
5) Optimize local factor and model order:
   According to certain local factor and model order, do
   5.1) Determine neighborhood dataset: Determine neighborhood dataset based on local factor
   5.2) Determine kernel weights: For each point in validation set, determine weights through kernel function and distance matrix
   5.3) Multivariate low-order approximation: Construct multivariate low-order model based on neighborhood dataset and given model order
   5.4) Evaluate learning effect: On validation set, calculate model's $R^2$, larger $R^2$ means better effect
   5.5) Record the best local factor and model order for prediction
   5.6) Tuning: Adjust local factor and model order within [0.05, 0.5] and {0, 1, 2}, respectively repeat this step until optimal local factor and model order are reached
6) Calculate prediction metrics: Based on test set, calculate model's $R^2$
7) Model fine-tuning: Under optimized local factor and model order, use all data to fine-tune model coefficients
8) Prediction: Calculate predicted results of pending dataset under fine-tuned model
9) Output: Inverse transform predicted results, print out inverse transformed results, and output prediction metrics ($R^2$)

## 2.3. Algorithm flowchart

The algorithm flowchart for multivariate local learning with embedded kernel functions (i.e., KELL) is shown in Figure 1.

## 2.4. Kernel functions

KELL can adopt any of the following five (or other) kernel functions:

(1) Sigmoid kernel

$$K(x_i, x_j) = \frac{2\pi}{e^{-\|x_i - x_j\|} + e^{\|x_i - x_j\|}}$$

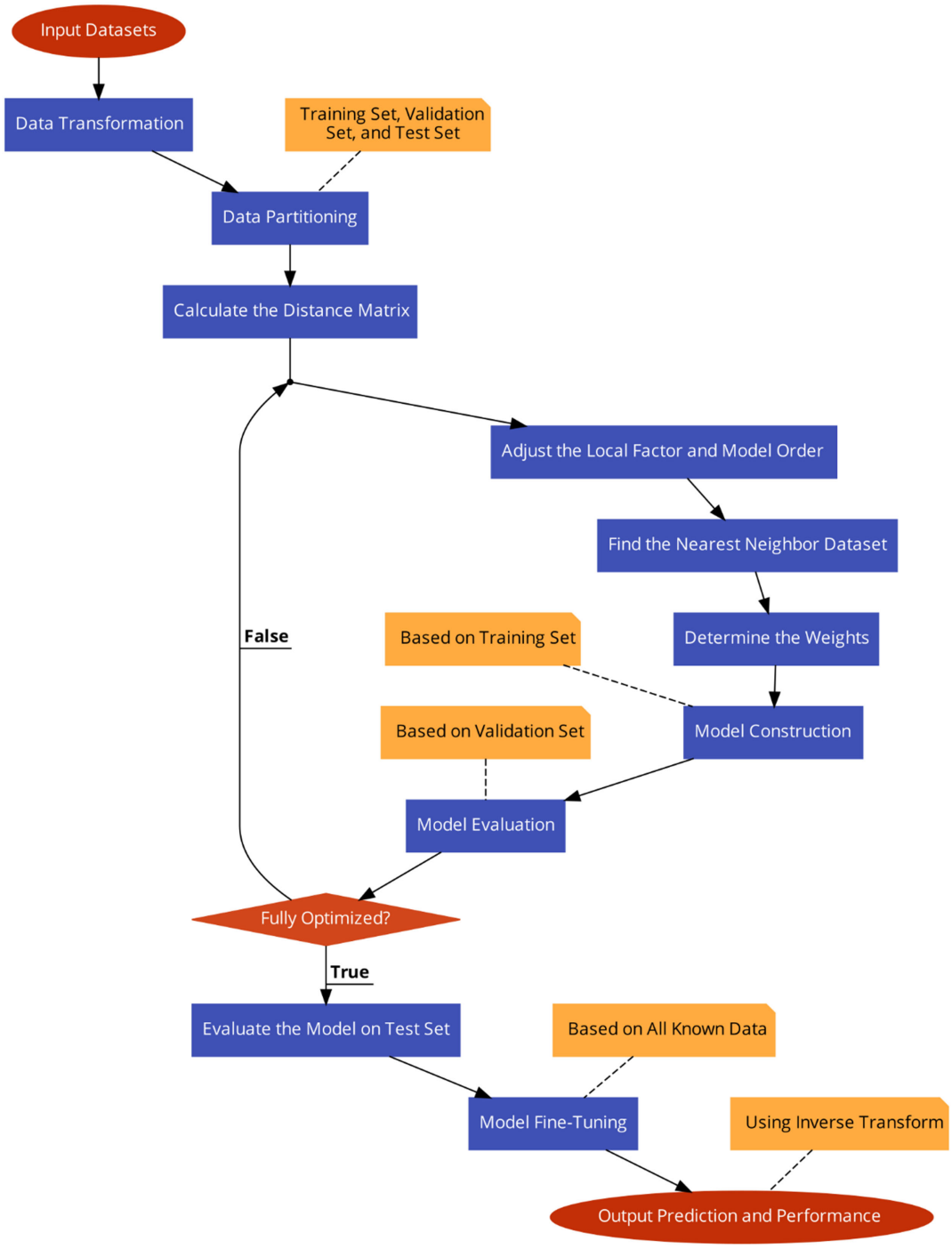
(2) Gauss kernel

$$K(x_i, x_j) = \frac{1}{\sqrt{2\pi}} e^{-0.5\|x_i - x_j\|^2}$$

(3) Laplace RBF kernel

$$K(x_i, x_j) = \frac{1}{\sqrt{2\pi}} e^{-\sqrt{2}\|x_i - x_j\|}$$

**Figure 1**
**Flowchart of kernel-based local learning**




(4) Logistic kernel

$$K(x_i, x_j) = \frac{1}{(1 + e^{-\|x_i - x_j\|}) + (1 + e^{\|x_i - x_j\|})}$$

(5) Silverman kernel

$$K(x_i, x_j) = \frac{1}{2} e^{-\frac{\|x_i - x_j\|}{\sqrt{2}}} \sin\left(\frac{\|x_i - x_j\|}{\sqrt{2}} + \frac{\pi}{4}\right)$$

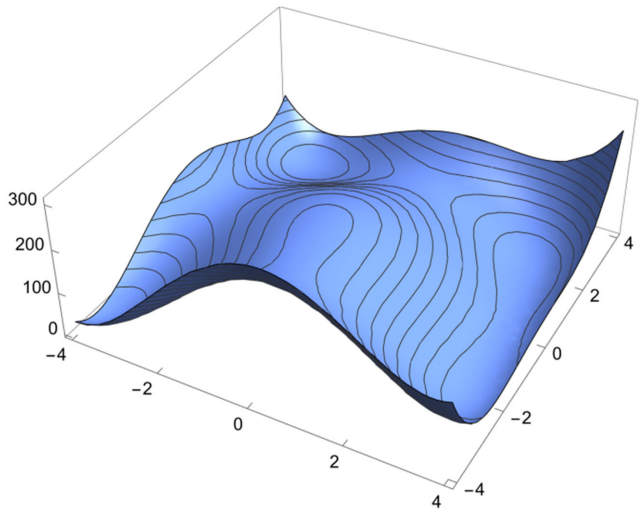
## 3. Numerical Results

### 3.1. Test problems and model performance

To verify the effectiveness of the KELL method, we first take a set of synthetic data as an example to test the performance of the KELL method in establishing complex nonlinear models and the influence of its parameters.

The synthetic data use the Himmelblau's function (Houssein et al., 2021) (Figure 2) as the target model. Using 2023 as the random seed, 100, 1000, 2000, 5000, and 10,000 training samples (including training set and validation set) are randomly generated uniformly in the spatial subset $[-4, 4]\times[-4, 4]$ of $\mathbb{R}$ , respectively, for the learning and adaptive training of the model. Another 100 test samples (test set) are randomly generated to test the performance of models obtained by various machine learning methods. The main metrics used to judge model performance are root mean square error (RMSE), relative RMSE, coefficient of determination ($R^2$), and $\log(1-R^2)$.

**Figure 2**
**Landscape of Himmelblau's function**




### 3.2. Influence of KELL algorithm parameters

It is noted that KELL mainly contains three hyperparameters: kernel function, local factor, and model order. Among them, the kernel function has little influence on KELL's performance (see comparison between Figure 3(a) and (c), Figure 3(b) and (d)); thus, it is ignored in algorithm optimization (Figure 1). In the following test comparisons, the Gaussian kernel function is used as the kernel function.

The influence of local factor and model order is demonstrated below.

Comparing Figure 3(a) and (b), Figure 3(c) and (d), it can be seen that using different model performance parameters (such as relative RMSE and $\log(1-R^2)$ in the figures) leads to consistent evaluation results.

Model order and local factor both have significant influence on the model. Thus, they are important hyperparameters of the KELL algorithm (Figure 1).

### 3.3. Comparison between KELL and Kernel methods

To evaluate the performance of the KELL method, we first compared the memory requirements of KELL and typical kernel methods represented by GPR (Fairbrother et al., 2022) (Table 1). The results show that the memory usage of GPR increases exponentially with the increase of sample data; the memory usage of KELL method increases linearly with the increase of sample data. For the same amount of data, the memory required by KELL is much less than the memory requirements of kernel methods like GPR.

**Table 1**
**Comparison of Memory Requirements between Kernel-Embedded Local Learning (KELL) and Gaussian Process Regression (GPR)**

| | Memory required | |
|---|---|---|
| Number of samples | Kernel-embedded local learning | Gaussian process regression |
| 100 | 98.86 KiB | 237.41 KiB |
| 1000 | 469.27 KiB | 22.91 MiB |
| 2000 | 775.55 KiB | 91.60 MiB |
| 5000 | 1.65 MiB | 572.32 MiB |
| 10,000 | 3.13 MiB | 2.24 GiB |

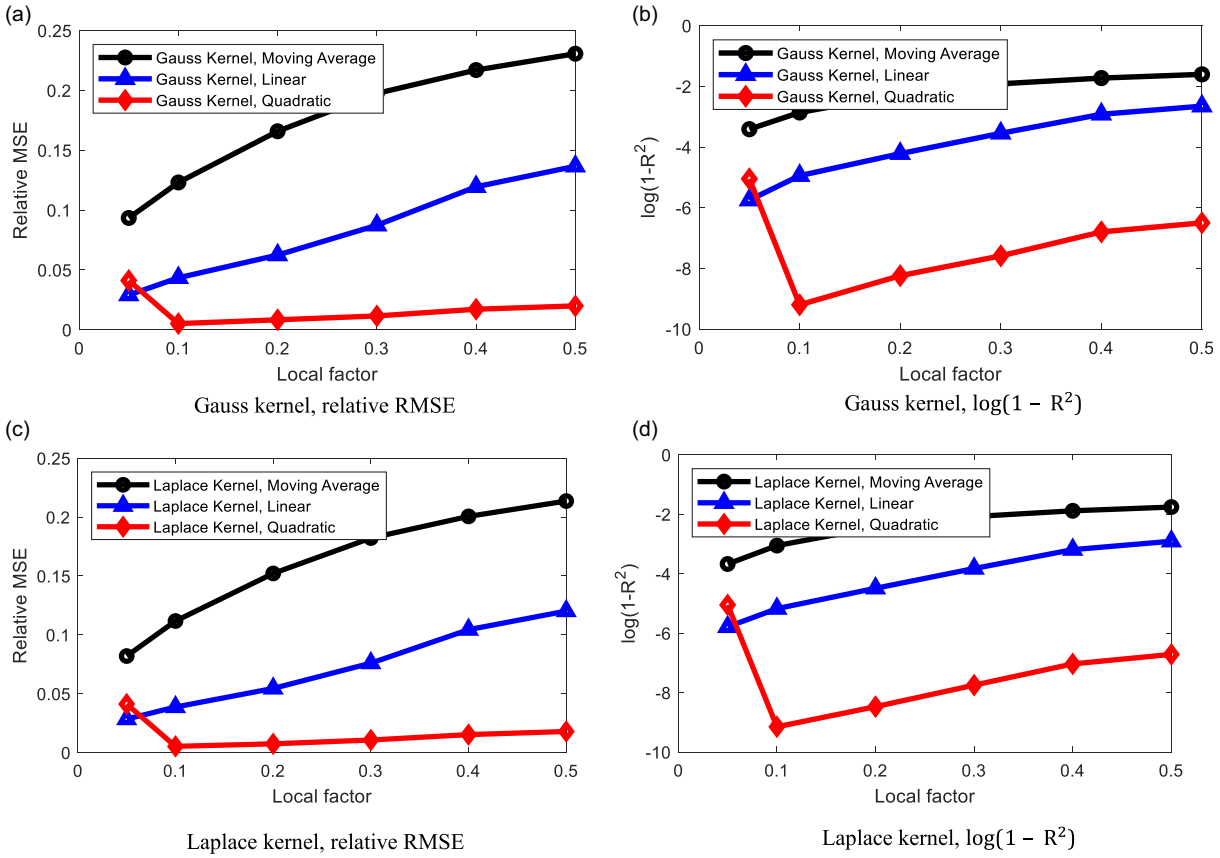
### 3.4. Comparison between KELL and different machine learning methods

Furthermore, we compare KELL with different machine learning methods (Table 2). Twenty-one machine learning methods were compared, including GPR (Liu et al., 2020), artificial neural networks (Schmidhuber, 2015), extra trees

**Table 2**
**Comparison of model accuracy between different machine learning methods**

| No. | ML methods | RMSE | $R^2$ |
|---|---|---|---|
| 1 | Kernel-based local learning | 0.9987 | 0.9998 |
| 2 | Gaussian process regression | 8.65779 | 0.95615 |
| 3 | Artificial neural networks | 19.6701 | 0.9511 |
| 4 | Extra trees regressor | 249.9066 | 0.8639 |
| 5 | Gradient boosting regressor | 329.7487 | 0.8216 |
| 6 | SVM regressor | 341.6532 | 0.8166 |
| 7 | Random forest regressor | 373.1002 | 0.8016 |
| 8 | K neighbors regressor | 494.4949 | 0.7272 |
| 9 | AdaBoost regressor | 564.0466 | 0.6778 |
| 10 | Decision tree regressor | 638.0672 | 0.6486 |
| 11 | Light gradient boosting machine | 1567.1942 | 0.1515 |
| 12 | Orthogonal matching pursuit | 1549.9326 | 0.1089 |
| 13 | Elastic net | 1669.8326 | 0.0701 |
| 14 | Lasso regression | 1721.1153 | 0.05 |
| 15 | Lasso least angle regression | 1699.4507 | 0.0494 |
| 16 | Bayesian ridge | 1722.6574 | 0.0489 |
| 17 | Ridge regression | 1728.5235 | 0.0478 |
| 18 | Least angle regression | 1737.6063 | 0.0436 |
| 19 | Linear regression | 1737.6065 | 0.0436 |
| 20 | Huber regressor | 1743.9313 | 0.0198 |
| 21 | Passive aggressive regressor | 1886.135 | −0.0689 |
| 22 | Dummy regressor | 2015.8817 | −0.1483 |

**Figure 3**
**The influence of different algorithm parameters on kernel-based local learning for the Himmelblau's problem: different kernels and different local factors**




Gauss kernel, relative RMSE

Gauss kernel, $\log(1 - R^2)$

Laplace kernel, relative RMSE

Laplace kernel, $\log(1 - R^2)$

regressor (Mastelini et al., 2022), gradient boosting regressor (Natekin & Knoll, 2013; Friedman 2001), SVM regressor (Rodríguez-Pérez & Bajorath, 2022), random forest regressor (Breiman, 2001), K-nearest neighbors regressor (Ertuğrul & Tağluk, 2017), AdaBoost regressor (Koduri et al., 2019), decision tree regressor (Pekel, 2020), light gradient boosting machine (Taha & Malebary, 2020), orthogonal matching pursuit (Zarei & Asl, 2021), elastic net (Zou & Hastie, 2005), lasso regression (Yazdi et al., 2021), lasso least angle regression (El Sheikh et al., 2021), Bayesian ridge regression (Yang & Yang, 2020), ridge regression (Arashi et al., 2021), least angle regression (Fernández-Delgado et al., 2019), linear regression (Hope, 2020), Huber regressor (Sun et al., 2020), passive aggressive regressor (Crammer et al., 2006), and dummy regressor (Schepers, 2016). The default parameters are used for each machine learning method (Ali, 2023).

It can be seen that compared to other methods, KELL has higher prediction accuracy.

## 3.5. Application of KELL in atmospheric data inversion

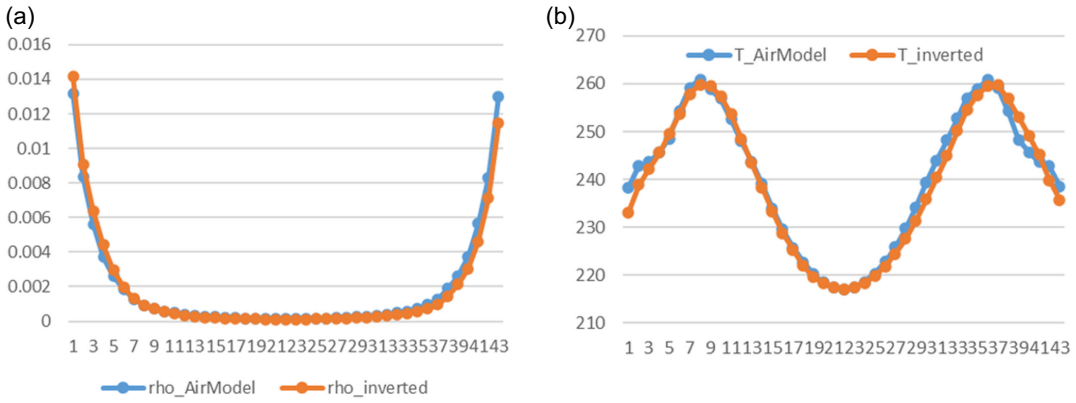
We used KELL to learn and model data derived from the direct numerical simulation of rarefied gas dynamics, which served as our training dataset. Subsequently, we used it to invert atmospheric data collected from a flight experiment or our pending dataset. This involved deducing the density and temperature of the atmosphere by analyzing the pressure ($p$) and heat flux ($q$) coefficients at various measurement locations on the aircraft's exterior. The training set comprised 87,365 sampling points and took into account 10 features, namely: $p_1$, $p_2$, $p_3$, $p_4$, $q_1$, $q_2$, and so on up to $q_6$. The pending dataset, which was to be inverted (or predicted) by KELL, consisted of only 43 points scattered along the flight path. A comparison with atmospheric models (as shown in Figure 4) revealed a high level of agreement, suggesting that KELL has successfully predicted the results.

## 4. Discussion

In the field of machine learning, dealing with complex nonlinear problems has always been an important challenge. Kernel methods achieve a linearization effect by introducing nonlinear transformations, thus achieving remarkable results in many problems. However, as data volume continues to increase, the memory pressure brought by kernel methods is also growing, which poses limitations on the application of algorithms. The KELL method proposed in this paper provides an effective approach to address this problem by combining kernel functions with local learning methods, achieving high prediction accuracy while reducing memory requirements. This will help promote the development of kernel methods in the field of data-intensive modeling and provide more possibilities for solving practical problems.

The key of the KELL method lies in embedding kernel functions into local learning methods and optimizing parameters like local factors and model orders. In practical applications, we can select appropriate kernel functions and corresponding local learning methods according

**Figure 4**
**Comparison of atmospheric local parameter inverted results and atmospheric model**




Density comparison

Temperature comparison

to the characteristics of different problems. In addition, parameter optimization is also an important feature of the KELL method. We can use methods like grid search and Bayesian optimization for parameter tuning to achieve optimal prediction performance.

It is worth noting that although the KELL method shows superiority in reducing memory requirements, there may be more cost in computation time. Unlike global learning methods such as Gaussian Process Regression (GPR), local learning methods require repeated modeling. For each prediction point, these methods sample a small batch of data based on its current location information. Consequently, the total computation may increase. However, in the case of data-intensive modeling, the limitation of memory resources is often more critical than computation time, so the KELL method still has high practical value.

In future research, we will further optimize the KELL method and explore more combinations of kernel functions and local learning methods. At the same time, we will also try to apply the KELL method to other fields such as image recognition and natural language processing to verify its generalization ability in different scenarios. In addition, we will also study how to implement the KELL method in a distributed computing environment to make full use of existing computing resources and improve algorithm efficiency.

## 5. Conclusions

This paper proposes a KELL method that achieves high prediction accuracy while reducing memory requirements by combining kernel functions with local learning methods. We conducted experiments on synthetic data and real datasets. The results show that compared with kernel methods like GPR, the KELL method has lower memory requirements when dealing with complex nonlinear problems; compared with other non-kernel methods, it has higher prediction accuracy. This provides strong support for the application of the KELL method in data-intensive modeling scenarios.

Although the KELL method has achieved significant results in reducing memory requirements, computation time may need further optimization. Future research can focus on how to reduce computation time while maintaining prediction accuracy to improve the applicability of the algorithm. In addition, we will also explore more combinations of kernel functions and local learning methods to expand the application scope of the KELL method to different problems and fields.

The proposal of the KELL method provides an effective approach to solving nonlinear problems in data-intensive modeling scenarios, combining the advantages of kernel methods and local learning methods. By optimizing parameters like local factors and model orders, the KELL method significantly reduces memory requirements while maintaining high prediction accuracy. This will help promote the development of kernel methods in the field of data-intensive modeling and provide more possibilities for solving practical problems.

The successful application of the KELL method will bring new opportunities to the field of machine learning, especially when facing complex nonlinear problems in data-intensive modeling scenarios. We believe that through continuous optimization and expansion of the KELL method, more breakthroughs can be achieved in the future, making greater contributions to high-precision prediction of complex nonlinear systems.

## Funding Support

This work was supported by the National Natural Science Foundation of China (Grant No. 12072353).

## Conflicts of Interest

The author declares that he has no conflicts of interest to this work.

## Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

## References

Ali, M. (2023). *PyCaret3.0: An open source, low-code machine learning library in Python*. Retrieved from: https://www.pycaret.org

Arashi, M., Roozbeh, M., Hamzah, N. A., & Gasparini, M. (2021). Ridge regression and its applications in genetic studies. *PLos One*, *16*(4).

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *The Journal of Machine Learning Research*, *2*, 499–526.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Baek, J., Chen, J., Susuki, K. (2022). A neural network-enhanced reproducing kernel particle method for modeling strain localization. *arXiv Preprint: 2204.13821*.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. USA: Cambridge University Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer Y. (2006). Online passive-aggressive algorithms, *Journal of Machine Learning Research*, *7*(19), 551–585.

El Sheikh, A. A., Barakat, S. L., & Mohamed, S. M. (2021). New aspects on the modified group LASSO using the least angle regression and shrinkage algorithm. *Information Sciences Letters*, *10*(3), 527–536.

Ertuğrul, Ö. F., & Tağluk, M. E. (2017). A novel version of k nearest neighbor: Dependent nearest neighbor. *Applied Soft Computing*, *55*, 480–490.

Fairbrother, J., Nemeth, C., Rischard, M., Brea, J., & Pinder, T. (2022). GaussianProcesses.jl: A Nonparametric Bayes package for the Julia language. *Journal of Statistical Software*, *102*(1), 1–36.

Fernández-Delgado, M., Sirsat, M. S., Cernadas, E., Alawadi, S., Barro, S., & Febrero-Bande, M. (2019). An extensive experimental survey of regression methods. *Neural Networks*, *111*, 11–34.

Francis, D. P., & Raimond, K. (2021). Major advancements in kernel function approximation. *Artificial Intelligence Review 54*, 843–876.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The Annals of Statistics*, *36*(3), 1171–1220.

Hope, T. M. (2020). Linear regression. In A. Mechelli & S. Vieira (Eds.), *Machine learning*. (pp. 67–81). Academic Press.

Houssein, E. H., Gad, A. G., & Wazery, Y. M. (2021). Jaya algorithm and applications: A comprehensive review. In N. Razmjooy, M. Ashourian & Z. Foroozandeh (Eds.), *Metaheuristics and Optimization in Computer and Electrical Engineering,* (pp. 3–24). Springer Cham.

Koduri, S. B., Gunisetti, L., Ramesh, C. R., Mutyalu, K. V., & Ganesh, D. (2019). Prediction of crop production using AdaBoost regression method. *Journal of Physics: Conference Series*, *1228*(1).

Lawrence, N. D. (2009). The Gaussian process latent variable model. In *Gaussian processes* (pp. 133–165). USA: CRC Press.

Liu, H., Ong, Y. S., Shen, X., & Cai, J. (2020). When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4405–4423.

Mastelini, S. M., Nakano, F. K., Vens, C., & de Leon Ferreira, A. C. P. (2022). Online extra trees regressor. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(10), 6755–6767.

Muhlbaier, M. D., Topalis, A., & Polikar, R. (2009). Learn++. MF: A first step towards incremental learning in nonstationary environments. In *2009 International Joint Conference on Neural Networks*, 2615–2622.

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*, 21.

Pekel, E. (2020). Estimation of soil moisture using decision tree regression. *Theoretical and Applied Climatology*, *139*(3–4), 1111–1119.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45.

Qiu, J., Xu, H., Zhu, X., & Adjeisah, M. (2023). Localized simple multiple Kernel K-means clustering with matrix-induced regularization. *Computational Intelligence and Neuroscience*, *2023*.

Quiñonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, *6*, 1939–1959.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning*. USA: MIT Press.

Rodríguez-Pérez, R., & Bajorath, J. (2022). Evolution of support vector machine and regression modeling in chemoinformatics and drug discovery. *Journal of Computer-Aided Molecular Design*, *36*(5), 355–362.

Schepers, J. (2016). On regression modelling with dummy variables versus separate regressions per group: Comment on Holgersson et al. *Journal of Applied Statistics*, *43*(4), 674–681.

Schölkopf, B., & Smola, A. J. (2018). *Learning with Kernels: Support vector machines, regularization, optimization, and beyond*. UK: MIT Press.

Schölkopf, B., Smola, A., & Müller, K. R. (1999). Kernel principal component analysis. In B. Schölkopf, C. Burges & A. Smola (Eds.), *Advances in kernel methods: support vector learning*, (pp. 327–352). USA: MIT Press.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Shalev-Shwartz, S. (2012). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, *4*(2), 107–194.

Sun, Q., Zhou, W. X., & Fan, J. (2020). Adaptive Huber regression. *Journal of the American Statistical Association*, *115*(529), 254–265.

Taha, A. A., & Malebary, S. J. (2020). An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access*, *8*, 25579–25587.

Vapnik, V. (2013). *The nature of statistical learning theory*. USA: Springer New York.

Yang, Y., & Yang, Y. (2020). Hybrid prediction method for wind speed combining ensemble empirical mode decomposition and Bayesian ridge regression. *IEEE Access*, *8*, 71206–71218.

Yazdi, M., Golilarz, N. A., Nedjati, A., & Adesina, K. A. (2021). An improved lasso regression model for evaluating the efficiency of intervention actions in a system reliability analysis. *Neural Computing and Applications*, *33*(13), 7913–7928.

Zarei, A., & Asl, B. M. (2021). Automatic seizure detection using orthogonal matching pursuit, discrete wavelet transform, and entropy based features of EEG signals. *Computers in Biology and Medicine*, *131*.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, *67*(2), 301–320.

**How to Cite:** Luo, C. (2024). KELL: A Kernel-Embedded Local Learning for Data-Intensive Modeling. *Artificial Intelligence and Applications,* 2(1), 38–44. https://doi.org/10.47852/bonviewAIA32021381