User Manual for *iLearnPlus*

Zhen Chen[1,†], Pei Zhao[2,†], Chen Li[3,†], Fuyi Li[3,4,5], Dongxu Xiang[3,4], Yong-Zi Chen[6], Tatsuya Akutsu[7], Roger J. Daly[3], Geoffrey I. Webb[4], Quanzhi Zhao[1,8,*], Lukasz Kurgan[9,*] and Jiangning Song[3,4,*]

[1]Collaborative Innovation Center of Henan Grain Crops, Henan Agricultural University, Zhengzhou 450046, China, [2]State Key Laboratory of Cotton Biology, Institute of Cotton Research of Chinese Academy of Agricultural Sciences (CAAS), Anyang, 455000, China, [3]Monash Biomedicine Discovery Institute and Department of Biochemistry and Molecular Biology, Monash University, Melbourne, VIC 3800, Australia, [4]Monash Centre for Data Science, Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia, [5]Department of Microbiology and Immunology, The Peter Doherty Institute for Infection and Immunity, The University of Melbourne, Melbourne, Victoria, 3000, Australia, [6]Laboratory of Tumor Cell Biology, Key Laboratory of Cancer Prevention and Therapy, National Clinical Research Center for Cancer, Tianjin Medical University Cancer Institute and Hospital, Tianjin Medical University, Tianjin 300060, China, [7]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Kyoto 611-0011, Japan, [8]Key Laboratory of Rice Biology in Henan Province, Henan Agricultural University, Zhengzhou 450046, China, [9]Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

[†]These authors contributed equally to this work.

[*]To whom the correspondence should be addressed. Tel: +61-3-9902-9304; Email: Jiangning.Song@monash.edu;

Correspondence may also be addressed to Quanzhi Zhao, Tel: +86-0371-56990209; Email: qzzhaoh@henau.edu.cn, and Lukasz Kurgan, Tel: +1-804-827-3986; Email: lkurgan@vcu.edu.

Table of Contents

1. Introduction *3*
2. Installing and running iLearnPlus *3*
Installation 3
Running 4
3. The workflow of iLearnPlus *5*
4. The input format of iLearnPlus *6*
5. The iLearnPlus-Basic module *7*
Feature descriptor extraction 7
Feature analysis 9
Predictor construction 10
Building machine-learning pipelines 13
6. The iLearnPlus-Estimator module *16*
7. The iLearnPlus-AutoML module *21*
8. The iLearnPlus-LoadModel module *21*
9. Other functions *23*
10. Performance evaluation strategy in iLearnPlus *26*
11. Online web server *27*
12. Descriptions of feature descriptors for nucleotide sequences *33*
13. Descriptions of feature descriptors for protein or peptide sequences *52*
14. Guidance of parameters setting for conventional machine learning algorithms *74*
15. Guidance of parameters setting for deep learning algorithms *78*
16. Guide to source code location *79*
References *80*

1. Introduction

iLearnPlus is the first machine-learning platform with both graphical- and web-based user interface that enables the construction of automated machine-learning pipelines for computational analysis and predictions using nucleic acid and protein sequences. Four major modules, including *iLearnPlus-Basic*, *iLearnPlus-Estimator*, *iLearnPlus-AutoML*, and *iLearnPlus-LoadModel*, are provided in *iLearnPlus* for biologists and bioinformaticians to conduct customizable sequence-based feature engineering and analysis, machine-learning algorithm construction, performance assessment, statistical analysis, and data visualization, without additional programming. *iLearnPlus* integrates 21 machine-learning algorithms (including 12 conventional classification algorithms, two ensemble-learning frameworks and seven deep-learning approaches) and 19 major sequence encoding schemes (in total 147 feature descriptors), outnumbering all the current web servers and stand-alone tools for biological sequence analysis, to the best of our knowledge. In addition, the friendly GUI (Graphical User Interface) of *iLearnPlus* is available to biologists to conduct their analyses smoothly, significantly increasing the effectiveness and user experience compared to the existing pipelines. *iLearnPlus* is an open-source platform for academic purposes and is available at https://github.com/Superzchen/iLearnPlus/. The *iLearnPlus-Basic* module is online accessible at http://ilearnplus.erc.monash.edu/.

2. Installing and running *iLearnPlus*

Installation

iLearnPlus is an open-source Python-based toolkit, which operates in the Python environment (Python version 3.6 or above) and can run on multiple operating systems (e.g. Windows, Mac, and Linux). Prior to installing and running *iLearnPlus*, all the dependencies should be installed in the Python environment, including PyQt5, qdarkstyle, numpy (1.18.5), pandas (1.0.5), threading, sip, datetime, platform, pickle, copy, scikit-learn (0.23.1), math, scipy (1.5.0), collections, itertools, torch (≥1.3.1), lightgbm (2.3.1), xgboost (1.0.2), matplotlib (3.1.1), seaborn, joblib, warnings, random, multiprocessing, and time. For convenience, we strongly recommend users install the Anaconda Python environment in their local computers, which can be freely downloaded from

https://www.anaconda.com/. The detailed steps of installing these dependencies are provided as follows:

Step 1. Download and install the anaconda platform:

Download from: https://www.anaconda.com/products/individual

Step 2. Install PyTorch:

Please refer to https://pytorch.org/get-started/locally/ for PyTorch installation.

Step 3. Install lightgbm, xgboost and qdarkstyle:

pip3 install lightgbm

pip3 install xgboost

pip3 install qdarkstyle

Running

To run *iLearnPlus*, go to the installation folder of *iLearnPlus* and run the 'iLearnPlus.py' script as follows:

python ilearnplus.py

Once *iLearnPlus* has started, the interface will show as demonstrated in **Figure S1**.

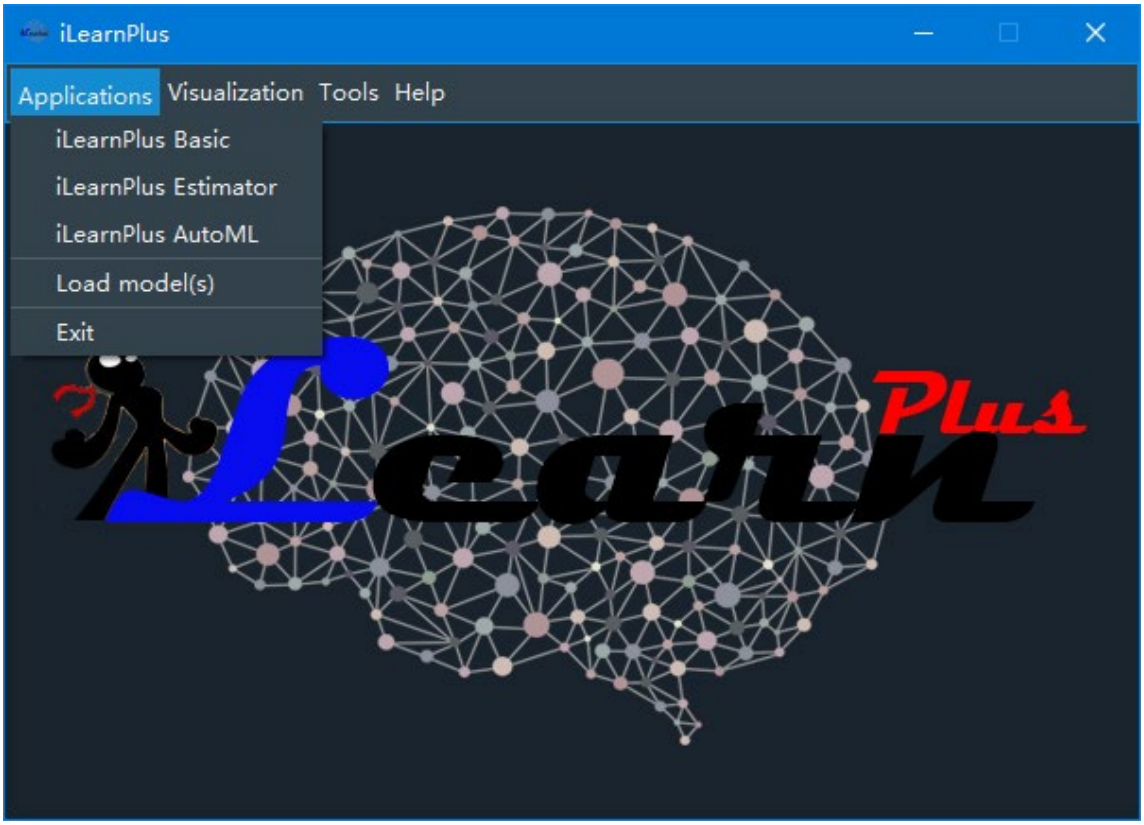


Figure S1. The main interface of the GUI version of *iLearnPlus*.

3. The workflow of *iLearnPlus*

Here we provide a step-by-step user instruction to demonstrate the workflow of *iLearnPlus* toolkit by running the examples provided in the "examples" directory. Five basic functions were designed and implemented in *iLearnPlus*, including feature extraction, feature analysis, predictor construction, and data/result visualization. Using these basic functions, four modules, including *iLearnPlus-Basic*, *iLearnPlus-Estimator* and *iLearnPlus-AutoML*, and *iLearnPlus-LoadModel* were further designed to facilitate sequence-based analysis and predictions on different levels of complexity (**Table S1, Figure S1**).

Table S1. Functions of the four major built-in modules in *iLearnPlus*.

Module	Function
iLearnPlus-Basic	1) Extraction of 147 different types of feature descriptors for DNA, RNA and protein sequences. 2) 20 feature analysis algorithms (ten feature clustering, five feature selection, three dimensionality reduction, and two feature normalization algorithms). 3) 21 machine-learning algorithms (12 conventional classification algorithms, two ensemble-learning frameworks and seven deep-learning approaches) 4) Data visualization (scatter plots for clustering and dimensionality reduction results, histogram and kernel density plot for data distribution, ROC and PRC for performance evaluation)
iLearnPlus-Estimator	1) Estimation of the prediction ability for the selected descriptors by providing a more flexible way of feature extraction and calculation. 2) Data visualization (boxplot for the evaluation metrics of the *K*-fold cross-validation, heatmap for displaying the correlation or *p*-values matrix of the models, ROC and PRC curve for performance evaluation) 3) The bootstrap test and student's *t*-test were used to compare the prediction performance difference.
iLearnPlus-AutoML	1) Automated performance benchmarking of different machine-learning algorithms based on the input features. 2) Data visualization (boxplot for the evaluation metrics of the *K*-fold cross-validation, heatmap for displaying the correlation or *p*-values matrix of the models, ROC and PRC curve for performance evaluation) 3) The bootstrap test and student's *t*-test were used to compare the prediction performance difference.
iLearnPlus-LoadModel	1) Performing prediction using the generated models and testing dataset.

4. The input format of *iLearnPlus*

The input of *iLearnPlus* is a set of DNA, RNA or protein sequences in FASTA format with a specially designed header. The FASTA header consists of three parts: part 1, part 2 and part 3, which are separated by the symbol "|" (**Figure S2**). Part 1 is the sequence name while part 2 is the sample category information, which can be filled with any integer. For instance, users may use 1 to indicate the positive samples and 0 to represent the negative samples for a binary classification task, or use 0, 1, 2, … to represent different classes in multiclass classification tasks. Part 3 indicates the role of the sample, for example "training" would indicate that the corresponding sequence would be used as the training set for *K*-fold validation test, and "testing" indicates that the sequence would be used as the independent testing sample for independent testing.

For feature analysis and predictor construction, four file formats are supported, including LIBSVM (1) format, Comma-Separated Values (CSV), Tab Separated Values (TSV), and Waikato Environment for Knowledge Analysis (WEKA) (2) format. For LIBSVM, CSV and TSV format, the first column must be the sample label. Please find the "data" directory of the software for examples of these file formats.

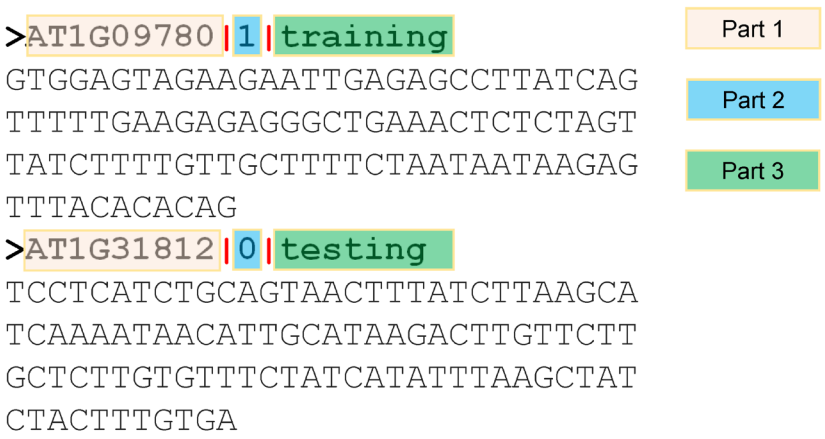


Figure S2. An example of the FASTA-formatted input DNA sequences used in *iLearnPlus* for feature descriptor extraction.

5. The *iLearnPlus-Basic* module

The *iLearnPlus-Basic* module aims at simply analysis and prediction using one protein/RNA/DNA descriptor and a machine-learning algorithm of choice. This module is particularly instrumental when interrogating the contributions of a certain type of feature descriptor or a specific machine-learning algorithm to the prediction performance. All the five basic functions including feature extraction, feature analysis, predictor construction, and data/result visualization can be implemented through the *iLearnPlus-Basic* module. There are four panels in *iLearnPlus-Basic* module. The "Descriptor" panel is used to extract feature descriptors for DNA, RNA and protein sequences, while the "Cluster / Dimensionality Reduction" and "Feature Normalization / Selection" panels are designed to implement the feature analysis algorithms, and the "Machine-learning" is used to build the prediction model.

Feature descriptor extraction

Each type of feature descriptor can be calculated using the "Descriptor" panel in the *iLearnPlus-Basic* module. Taking the DNA "DAC" descriptor as an example (**Figure S3**):

Step 1: Open the sequences file

Click the "Open" button in "Descriptor" panel and select the DNA sequences file (e.g. "DNA_sequences.txt" in "data" directory of *iLearnPlus* package). The biological sequence type (i.e. DNA, RNA or protein) will then be automatically detected based on the input sequences.

Step 2: Select the feature descriptor and configure the descriptor parameters

Click the "DAC" descriptor and set the corresponding parameters with the parameter dialog box (the default parameters were used here (**Figure S4**)). The information including sequence type, selected descriptor and the descriptor parameter(s) will be displayed in the "Parameters" area.

Step 3: Run the program

Click the "Start" button to calculate the descriptor features. The feature encoding and graphical presentation will be displayed in the "Data" and "Data distribution" panels, respectively. Here,

we used the histogram and kernel density plot to display the distribution of the feature encoding.

Step 4: Save the results and plots

Click the "Save" button to save the generated feature encodings. *iLearnPlus* supports four formats for saving the calculated features, including LIBSVM, CSV, TSV, and WEKA format, so as to facilitate direct use of the features in the following analysis, prediction model construction and the third-party computational tools, such as scikit-learn and WEKA. In addition, *iLearnPlus* provides the TSV_1 format, which includes the sample and feature labels. All the plots in *iLearnPlus* are generated by the matplotlib library and can be saved to a variety of image formats, such as PNG, JPG, PDF, TIFF etc).

Figure S3. An example of extracting feature descriptor using the *iLearnPlus-Basic* module.

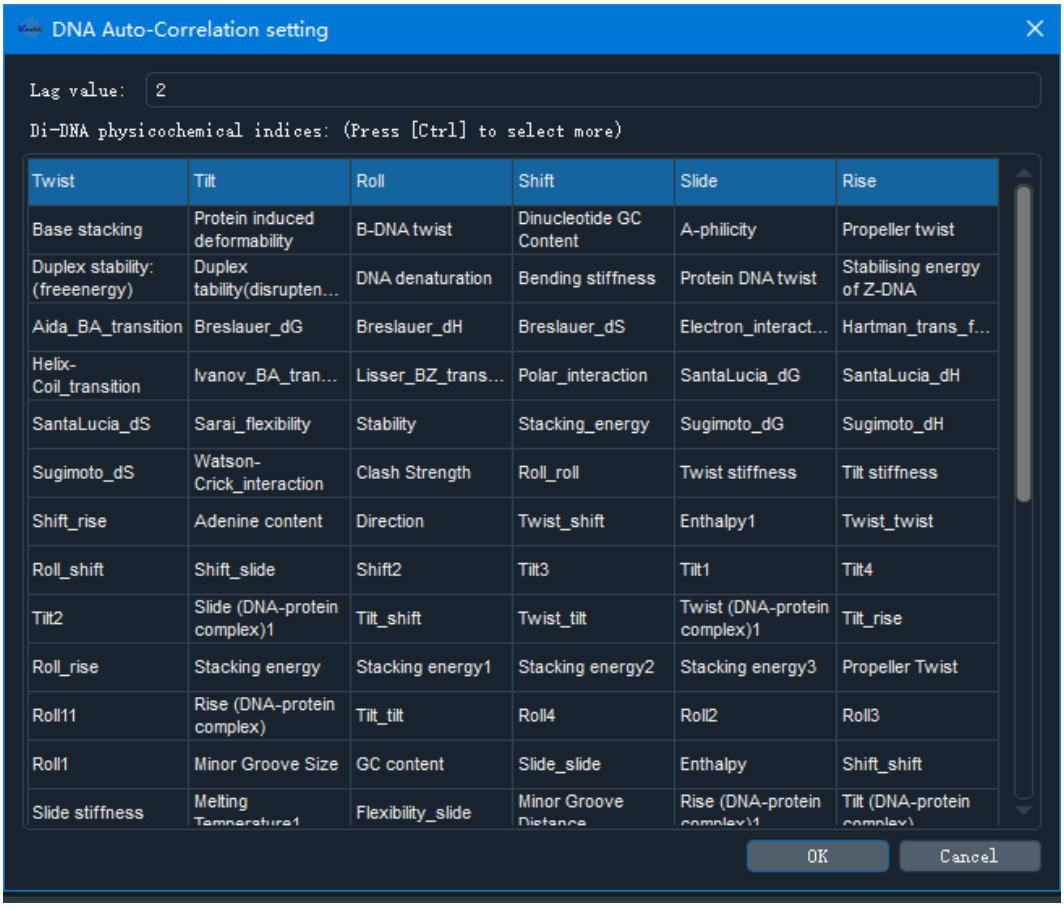


Figure S4. The dialog box for DNA DAC descriptor the parameter setting.

Feature analysis

iLearnPlus provides multiple options to facilitate feature analysis, including ten feature clustering, three dimensionality reduction, two feature normalization and five feature selection approaches (**Table 3** in our paper). In the *iLearnPlus-Basic* module, the "Cluster / Dimensionality Reduction" panel is used to deploy the clustering and dimensionality reduction algorithms; while the "Feature Normalization / Selection" panel is designed to implement the feature normalization and selection function. Taking the clustering as an example:

Step 1: Load data

There are two ways to load the data for analysis: 1) open a coding file and 2) select the data generated from other panels. Here we load the data from file. Click the "Open" button and open the "data.csv" in the "data" directory.

Step 2: Select the analysis algorithm and set the corresponding parameter(s)

Select “kmeans” clustering algorithm and set the cluster number as 5.

Step 3: Run the program

Click the “Start” button to start the analysis progress. The clustering result and graphical presentation will be displayed in the “Result” and “Scatter plot” panels, respectively. Here, we used the scatter plot to display the clustering result.

Step 4: Save the result and plot

Click the “Save” button to save the generated clustering results (**Figure S5**).

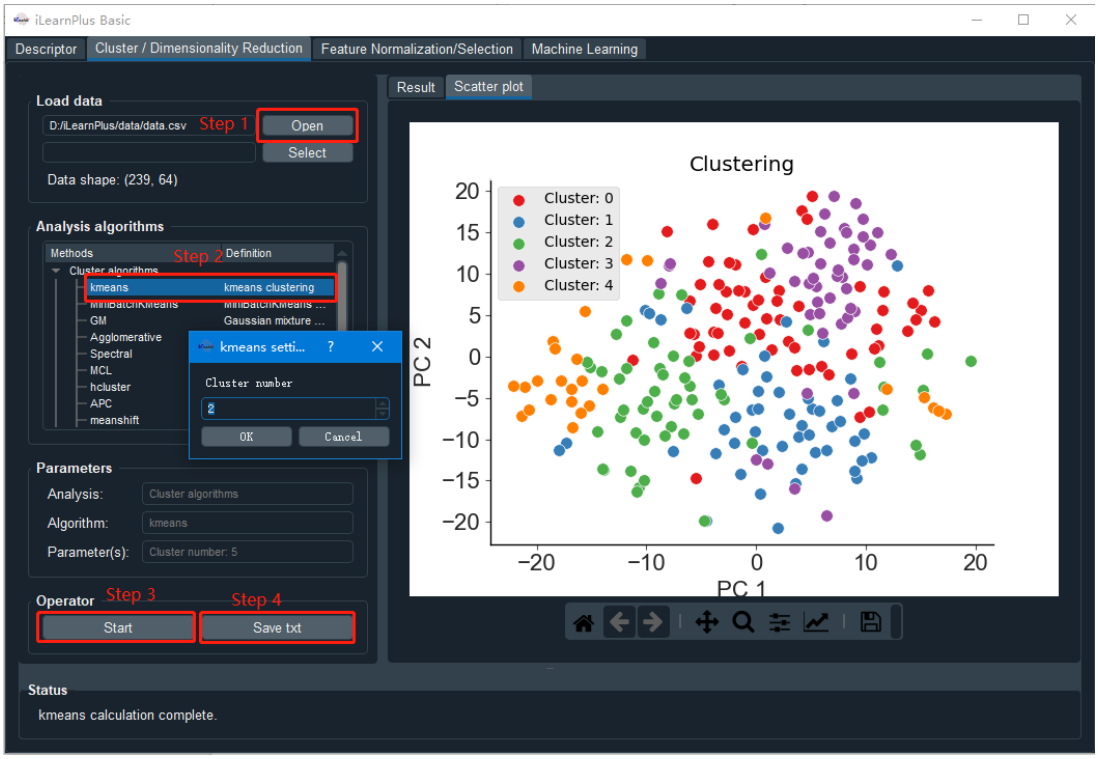


Figure S5. An example of implement the clustering algorithm using *iLearnPlus-Basic* module.

Predictor construction

iLearnPlus offers 12 conventional classification algorithms, two ensemble-learning frameworks, and seven deep-learning approaches. **Figure S6** shows the architectures of the deep-learning approaches. The implementation of these algorithms in *iLearnPlus* is based on four third-party machine-learning platforms, including scikit-learn (3), XGBoost (4), LightGBM (5), and PyTorch (6). Taking the CNN algorithm as the core machine-learning algorithm:

Step 1: Load data.

There are also two ways to load the data for analysis: 1) open a coding file and 2) select the data generated from other panels. Here we load the data from the "Descriptor" panel. At first, open the "m1A_DNA_sequences.txt" in the "data" directory and select the "binary" descriptor. Click "Start" button to calculate the feature encoding. Then, switch to the "Machine-learning" panel and load data by clicking the "Select" button (**Figure S7**). Select "Descriptor data" in the data selection dialog box and click the "OK" button.

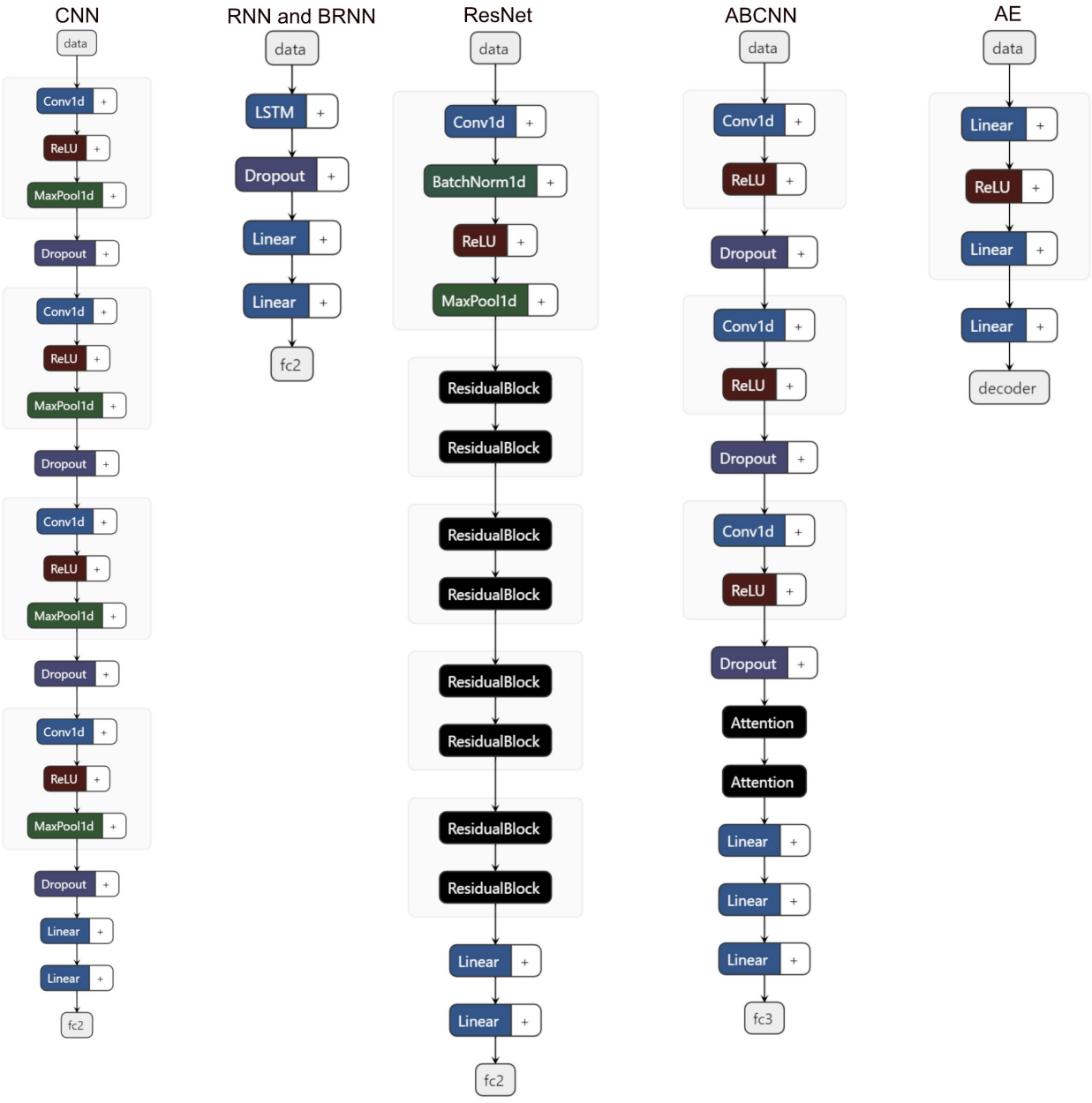


Figure S6. The architectures of the deep-learning approaches employed in *iLearnPlus*.

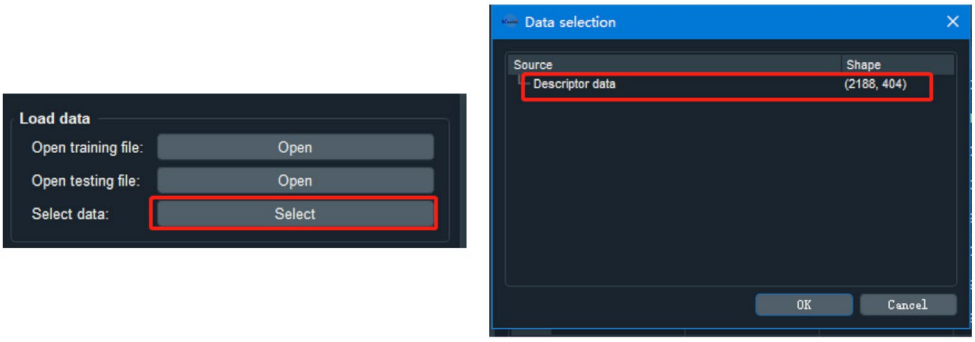


Figure S7. Loading data using the data selection explorer.

Step 2: Select the machine-learning algorithm and configure the corresponding parameter(s)

Select "Net_1_CNN" and set the "Input channels" as 4 (**Figure S8**). The default values were used for the remaining parameters.

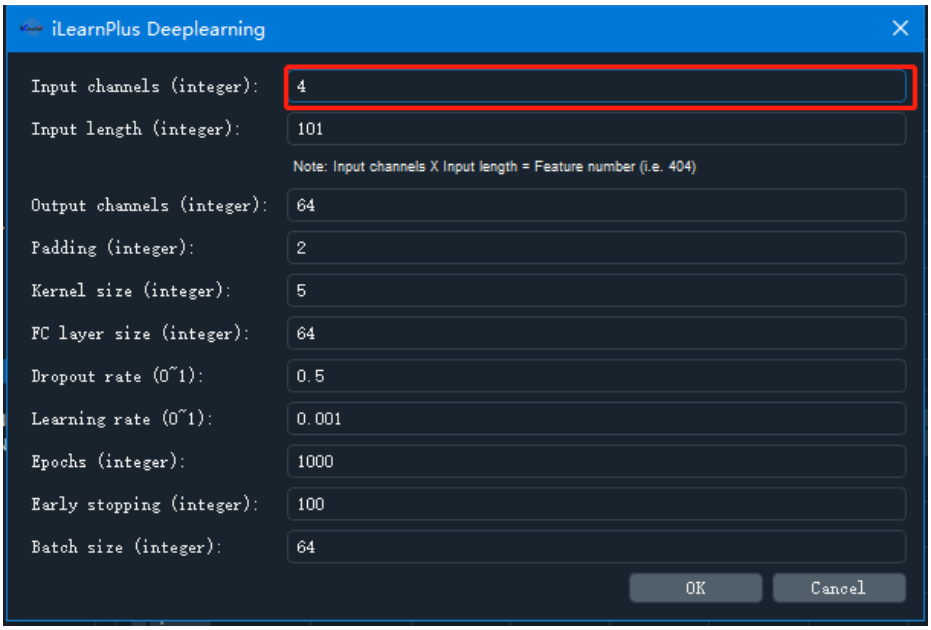


Figure S8. An example of parameter setting for the CNN algorithm.

Step 3: Set K-fold cross-validation

Set the *K* number as 5 (**Figure S9**).

Step 4: Run the program

Click the "Start" button to start the analysis progress. The prediction score, evaluation metrics for *K*-fold cross-validation test, independent test, and the ROC and PRC curve will be displayed

(**Figure S10**).

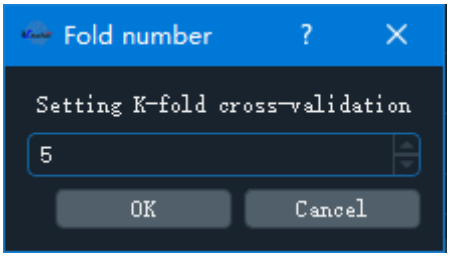


Figure S9. Setting five-fold cross-validation.

Figure S10. An example of model construction using CNN algorithm in *iLearnPlus*.

Building machine-learning pipelines

Usually, more than one individual functionality will be used in biological sequence analysis. In this case, the *iLearnPlus-Basic* module allows users to build their own machine-learning pipelines. The output data generated in the previous panel can be used as the input for next panel by using the graphical data selection explorer. Here, we take the malonylation site prediction as an example:

Step 1: Extract the descriptor using the iLearnPlus-Basic panel

In "Descriptor" panel, open the "Malonylation.txt" in "data" directory, select the ENAC

descriptor, and set the sliding window size as 8. Click “Start” button to calculate the descriptor (**Figure S11**).

Step 2: Select the top 100 features

Switch to the “Feature Normalization / Selection” panel and load the data which has been generated by the “Descriptor” panel, we can see that the data shape of the generated data is (10578, 480), indicating that there are 10578 samples and the dimension for the feature vector is 480. We used the CHI2 algorithm to select the top 100 features (**Figure S12**).

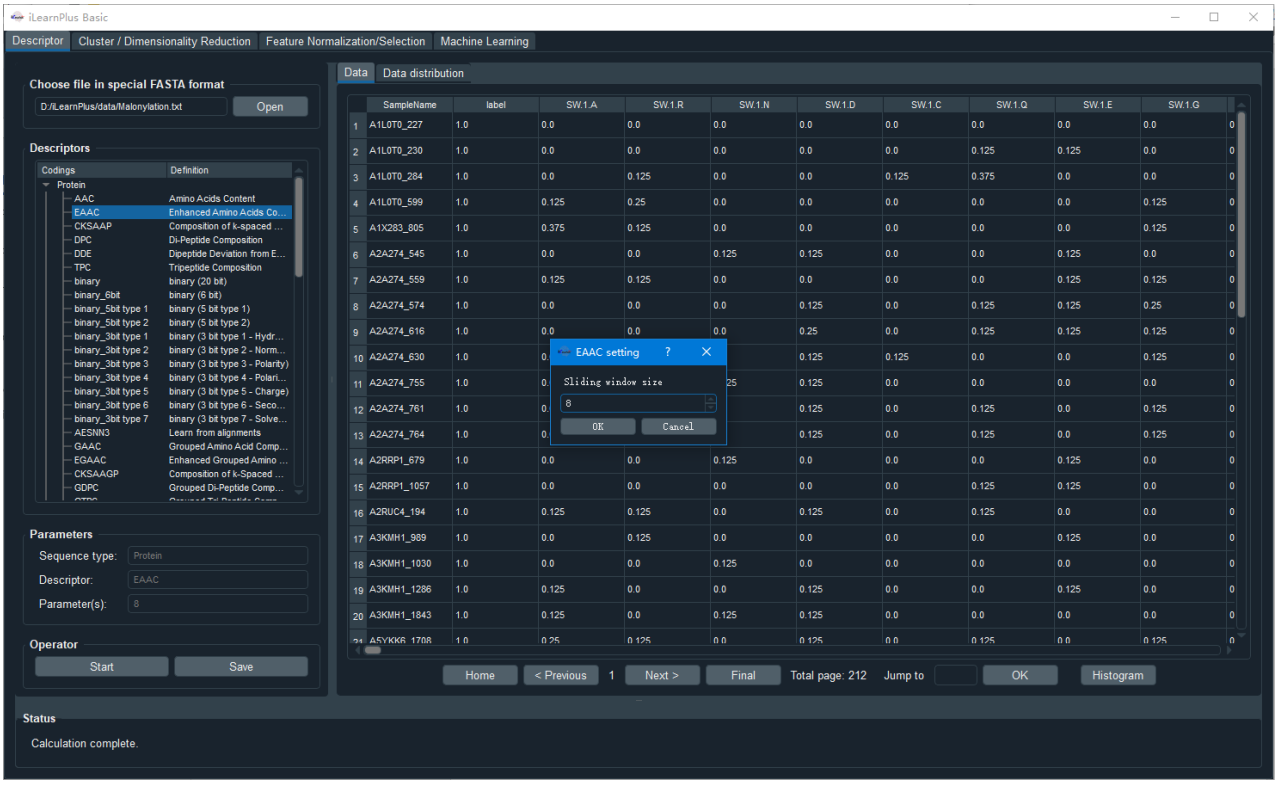


Figure S11. Extracting the EAAC descriptor using “Descriptor” panel in the *iLearnPlus-Basic* module.

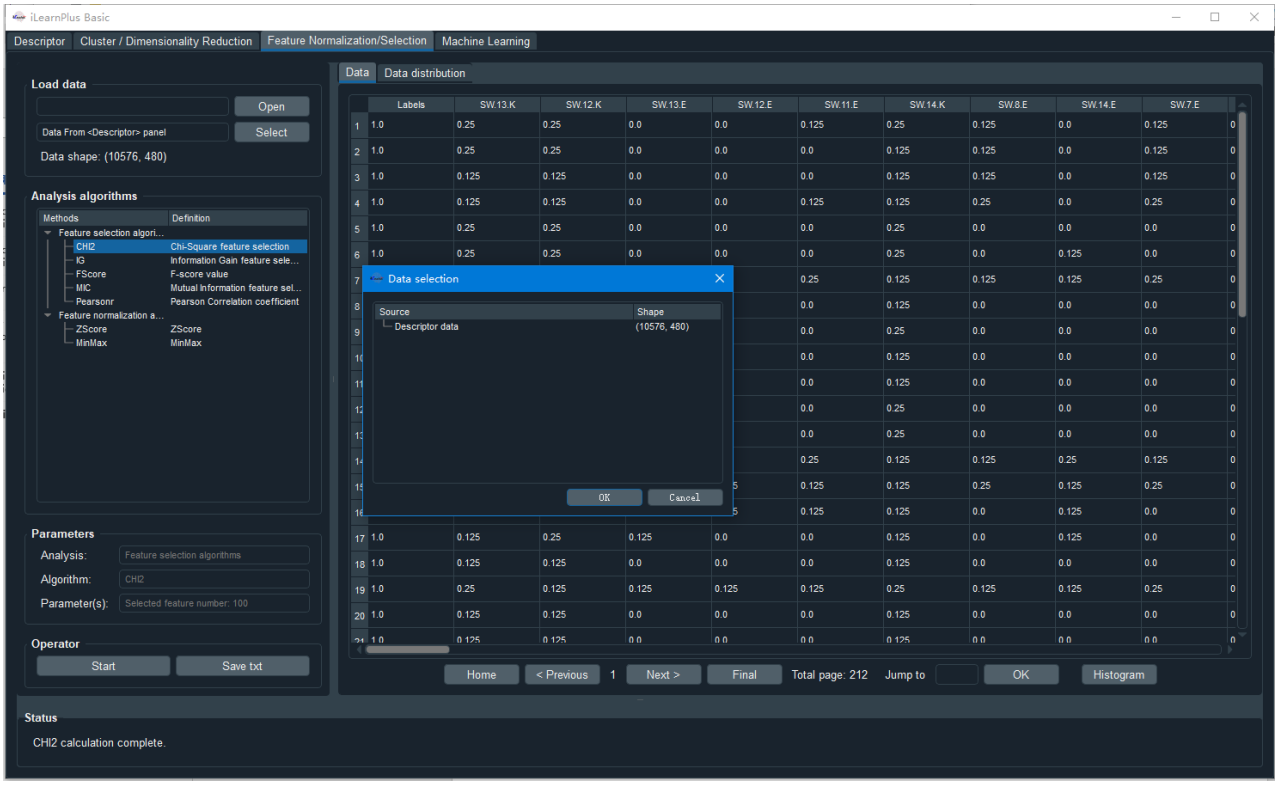


Figure S12. Selecting the top 100 features using "Feature Normalization/Selection" panel in the *iLearnPlus-Basic* module.

Step 3: Build the prediction model using the selected features

Switch to the "Machine-learning" panel, load the data from "Feature selection data" using the data selection dialog box in the "Machine-learning" panel and the data shape is (10576, 100) (**Figure S13**). Select the RF algorithm and use the default parameters to build the prediction model (**Figure S14**).

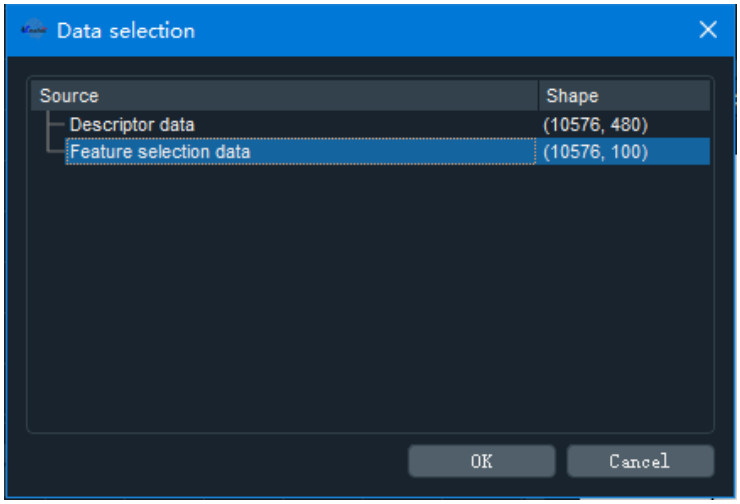


Figure S13. An example of loading data with the data selection dialog box.

Figure S14. An example of prediction model construction using RF algorithm in the *iLearnPlus-basic* module.

6. The *iLearnPlus-Estimator* module

The *iLearnPlus-Estimator* module provides a more flexible way of feature extraction and calculation by allowing users to select multiple feature descriptors of interest. For a prediction task, the *iLearnPlus-Estimator* module can select out the descriptor with best performance. Here, we take the m^1A site prediction as an example:

Step 1: Load sequence data

Open the "m1A_DNA_sequences.txt" file in "data" directory of the *iLearnPlus* package.

Step 2: Select the descriptors

Here, nine descriptors, including Kmer, NAC, ANF, ENAC, binary, KNN, Z_curve_9bit and MMI, were selected to evaluate their performance. The default parameters for the descriptors were used.

Step 3: Select machine-learning algorithm

The RF algorithm was selected to build prediction models, and the number of trees was set as

1000.

Step 4: Set K-fold cross-validation

Set the *K* as 5.

Step 5: Running the program

Click "Start" button to train the models. For each of the selected feature descriptors, the program will extract the feature encoding and build the prediction model automatically one by one (**Figure S15**).

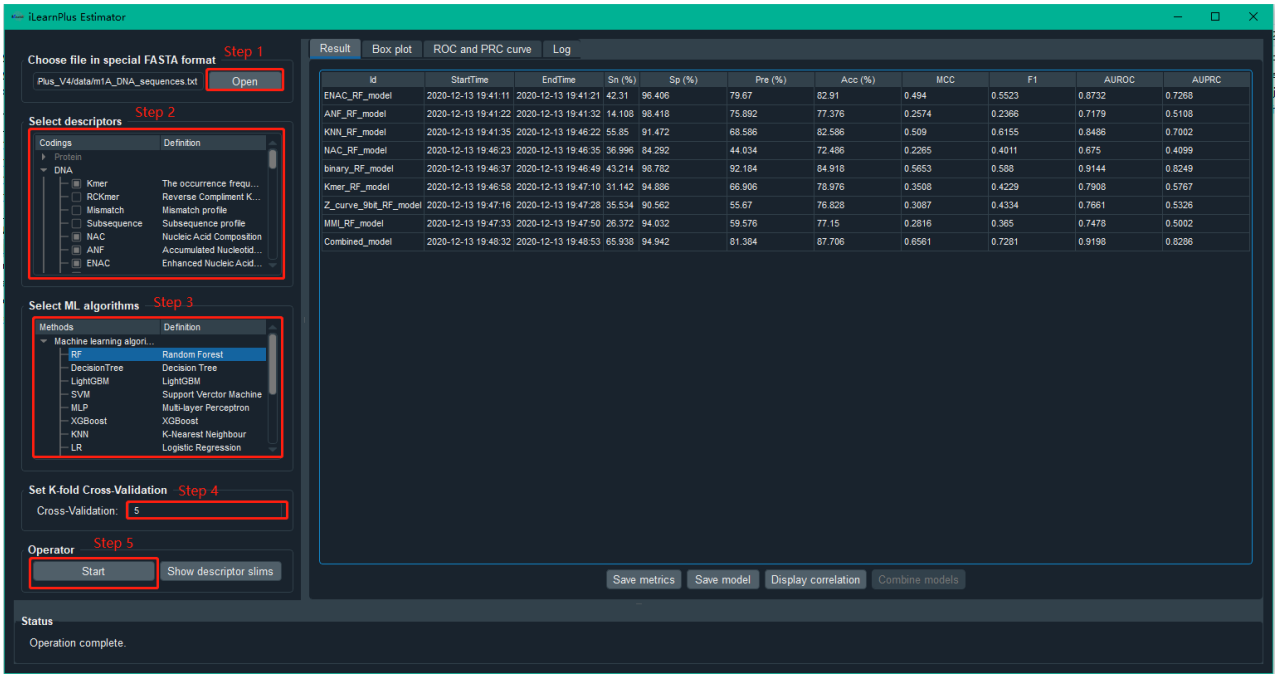


Figure S15. The panel of the *iLearnPlus-Estimator* module.

Step 6: Display prediction results

1) The evaluation metrics for the nine classifiers were displayed in the table widget (**Figure S15**).
2) The correlation matrix of the nine classifiers was displayed using the heatmap (**Figure S16**).
3) Boxplot for the evaluation metrics (**Figure S17**).
4) ROC and PRC curves (**Figure S18**).

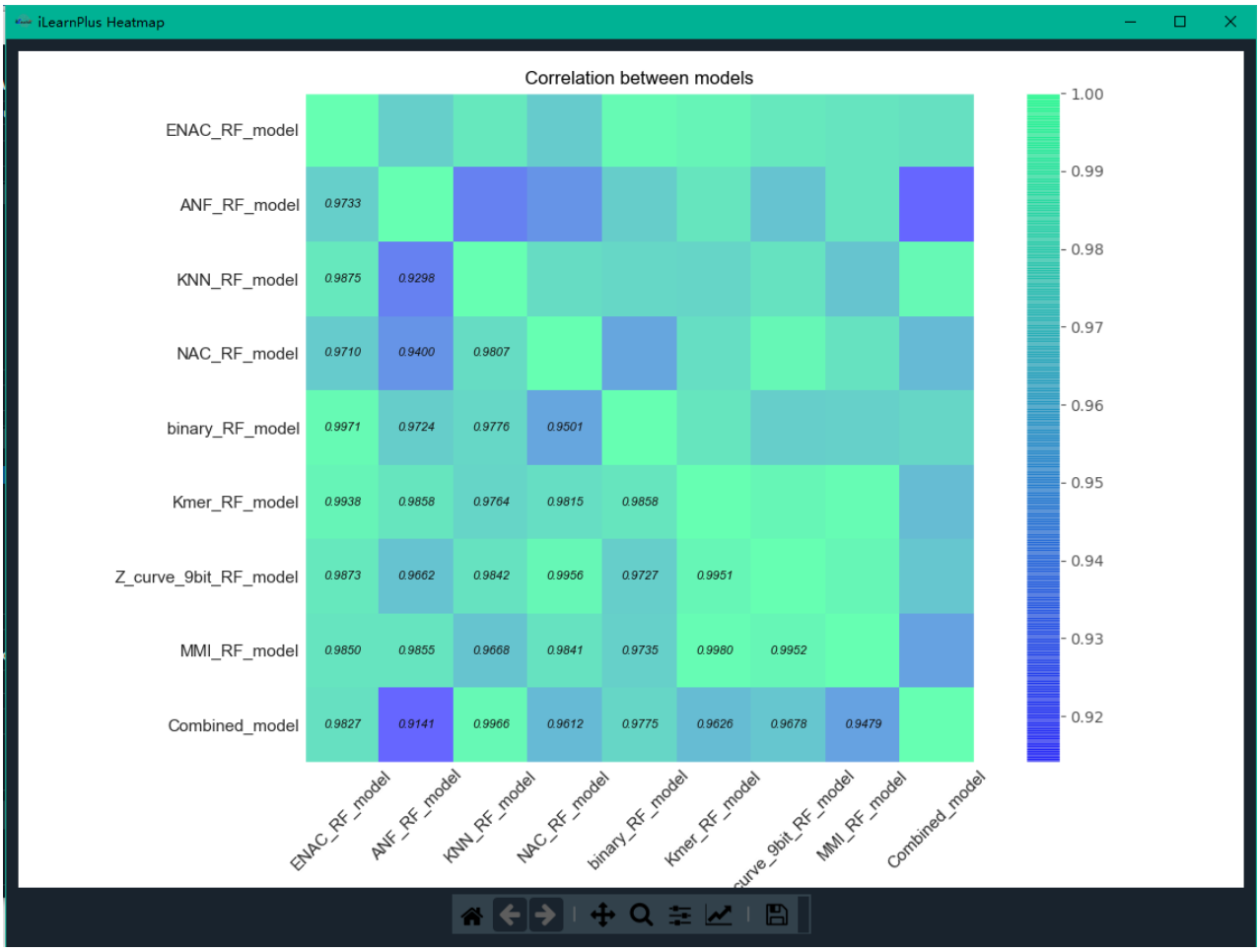


Figure S16. The correlation matrix generated by the *iLearnPlus-Estimator* module.

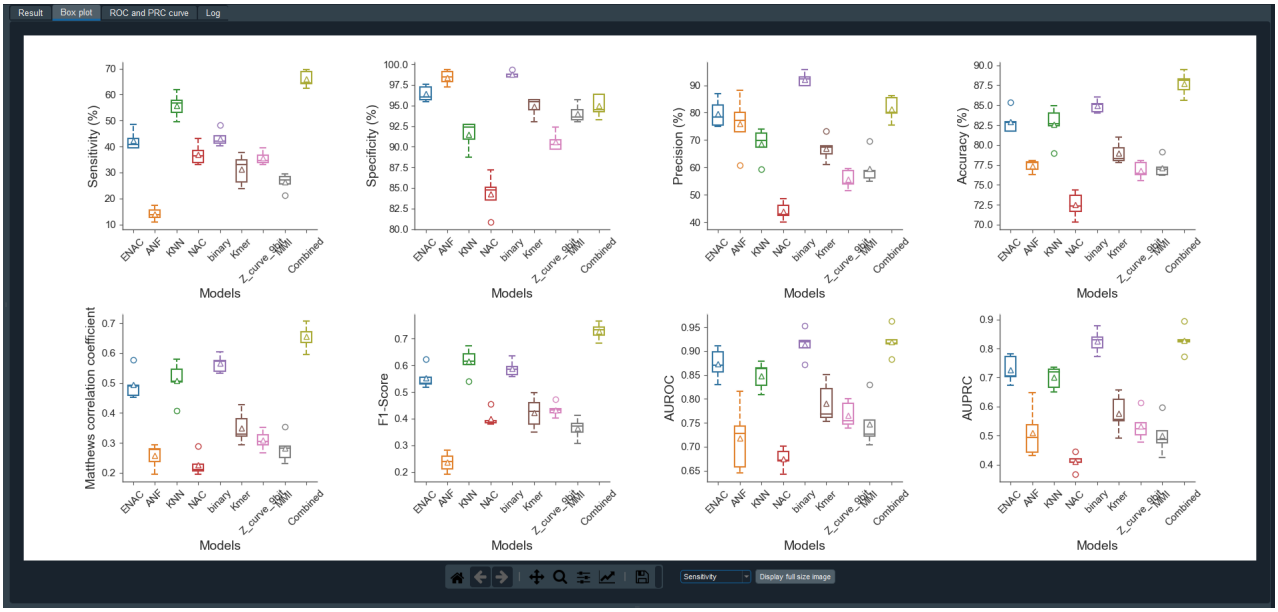


Figure S17. The boxplot generated by the *iLearnPlus-Estimator* module.

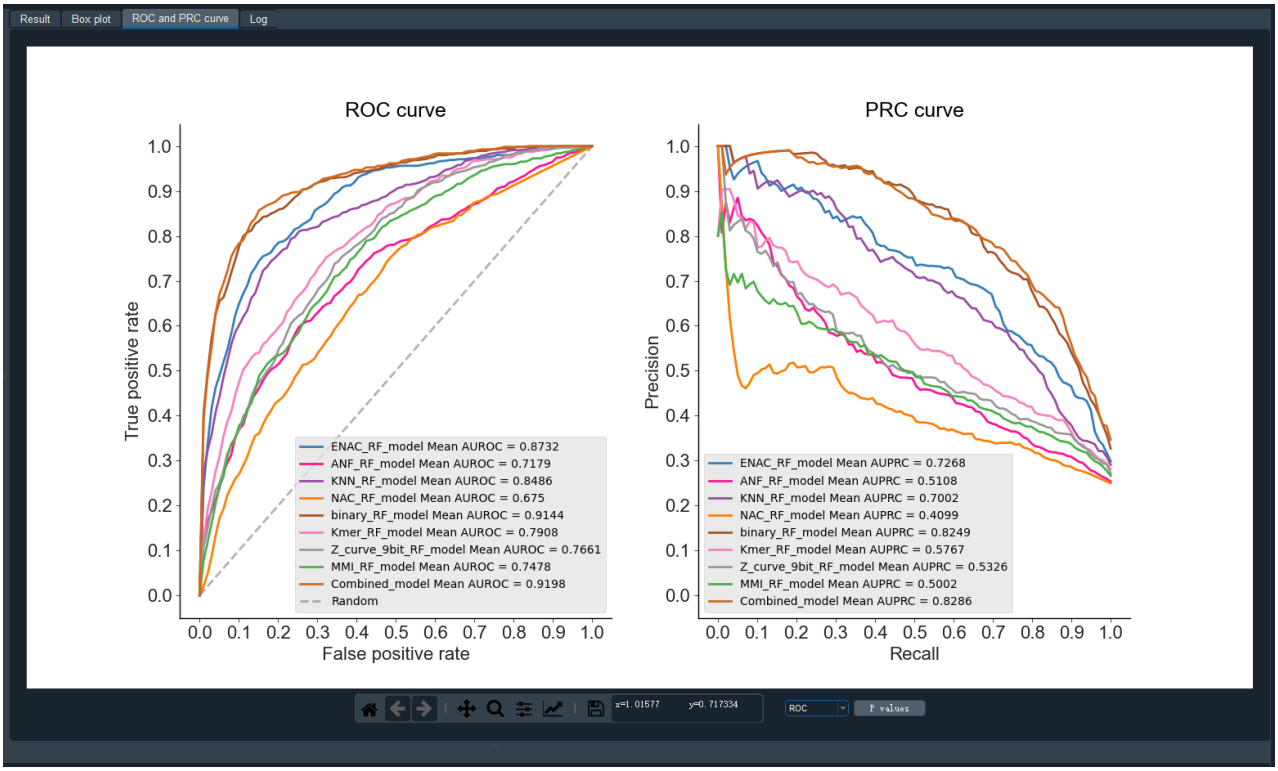


Figure S18. The ROC and PRC curves generated by the *iLearnPlus-Estimator* module.

In addition, *iLearnPlus* offers two statistical tests for users to compare the prediction performance difference. The student's *t*-test is used to statistically compare the means of any two performance evaluation measures (e.g. *Sn*, *Sp*, *Acc*, *MCC* etc.) obtained via the *K*-fold cross-validation test (**Figure S19**); while a bootstrap test was used to assess the significance of performance difference between all pairs in the ROC or PRC curve (**Figure S20**).

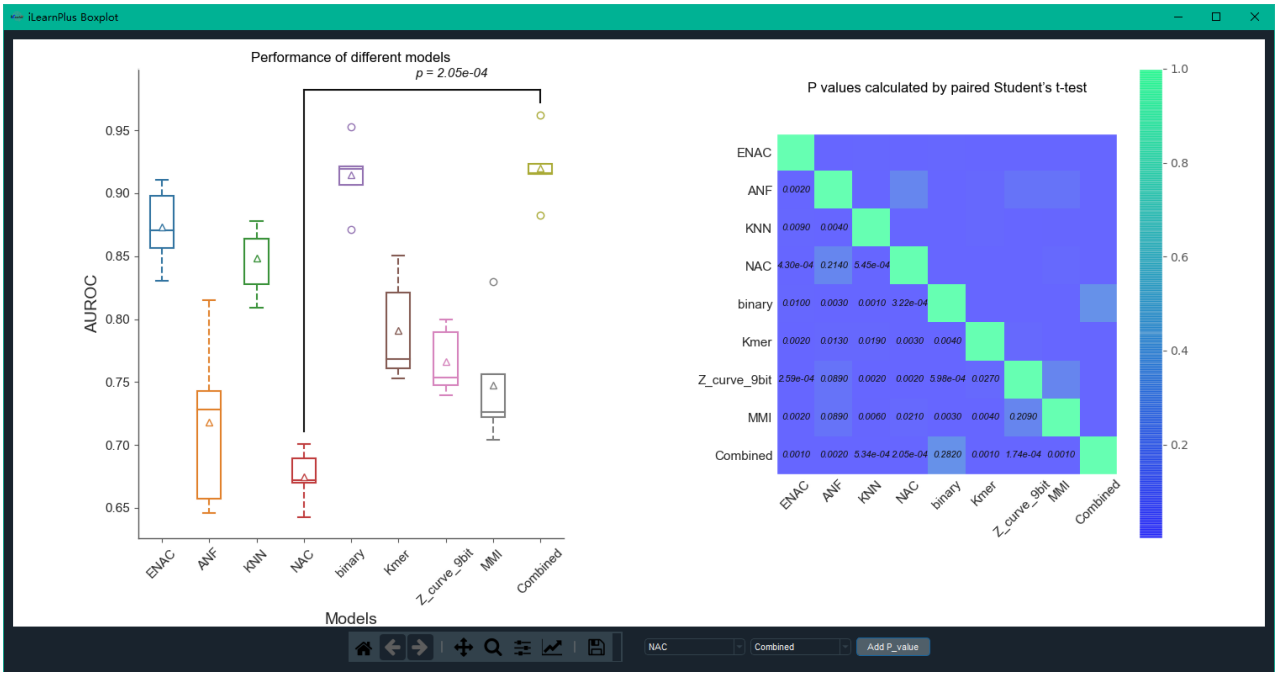


Figure S19. The paired *p*-values calculated by the student's *t*-test in the *iLearnPlus-Estimator* module.

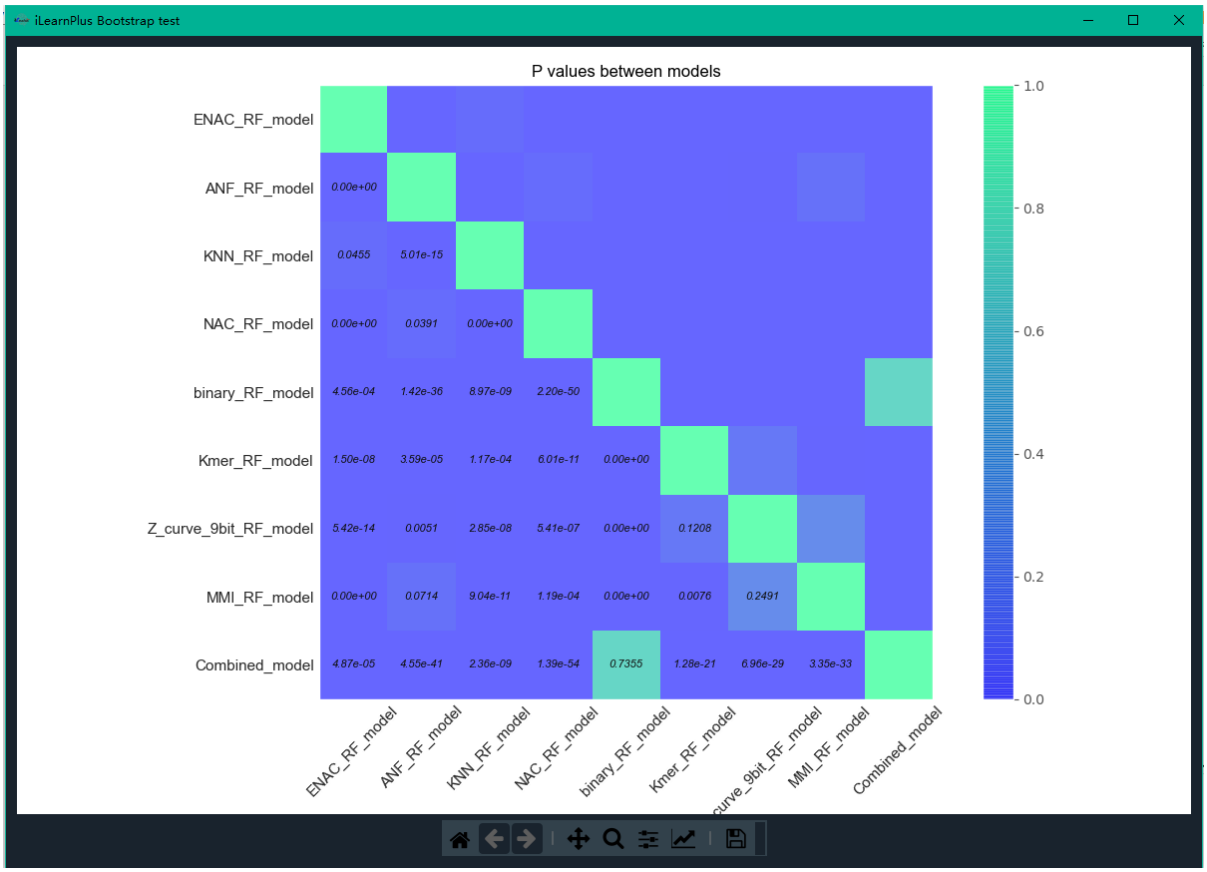


Figure S20. The paired *p*-values calculated by the bootstrap method in the *iLearnPlus-Estimator* module.

iLearnPlus-estimator and *iLearnPlus-AutoML* modules facilitate merging results produced by multiple machine-learning algorithms to train new models. This is implemented by combining the multiple predictions with the underlying goal to improve predictive performance. To do so, *iLearnPlus* supports automatic evaluation of the performance for all possible combinations of the selected models. Compared with iLearn, which only uses a logistic regression approach to integrate the classifiers, *iLearnPlus* provide more options to use different algorithms in order to integrate the trained models. User can click the "Combine models" button in the result panel. For example, the best performance of the eight descriptor is binary (AUROC=0.9144), while the combined model has achieved the best performance with AUROC=0.9198. The combined models include 'ENAC_RF_model', 'ANF_RF_model', 'KNN_RF_model', 'NAC_RF_model', 'binary_RF_model', 'Kmer_RF_model', 'Z_curve_9bit_RF_model', and 'MMI_RF_model', which can be found in log panel.

7. The *iLearnPlus-AutoML* module

The *iLearnPlus-AutoML* module focuses on automated performance benchmarking of different machine-learning algorithms based on the input features. The usage of the *iLearnPlus-AutoML* module is similar with the usage of the *iLearnPlus-Estimator* module. The difference of the two models is that the input of the *iLearnPlus-Estimator* module only contains biological sequences; while the input of the *iLearnPlus-AutoML* module is the feature encoding information in CSV, TSV, LIBSVM or WEKA format. Combining the *iLearnPlus-Estimator* and *iLearnPlus-AutoML* modules, users can evaluate and compare the prediction performance using all the selected feature descriptors and machine-learning algorithms in an efficient manner.

8. The *iLearnPlus-LoadModel* module

All the generated models can be saved ("*.pkl") in the aforementioned three modules and users can upload their models and testing dataset to perform prediction directly via the *iLearnPlus-LoadModel* module. The saved convetional machine-learning modules can be direclty applied indepdently in the scikit-learn running environment by invoking the joblib library to load the model. The exported deep-learning models, on the other hand, can be used with the help of PyTorch library independently,

with the prerequisite of importing the architectures of the neural network. To use the model in their own applications, advanced users can refer to the code in "iLearnPlusLoadModel.py" for the detail usage.

Step 1: Load models

Click the "Load" button and select the models in the "models" directory, one or more modules can be selected at one time.

Step 2: Load the testing file

Click the "Open" button and select the "binary_ind.csv" file in the "data" directory of the *iLearnPlus* package.

Step 3: Run the program

Click the "Start" button to predict the testing file using the loaded models. If multiple models are loaded, the average prediction score of the models will be displayed. In addition, the evaluation metrics, ROC and PRC curves will also be displayed (**Figure S21**).

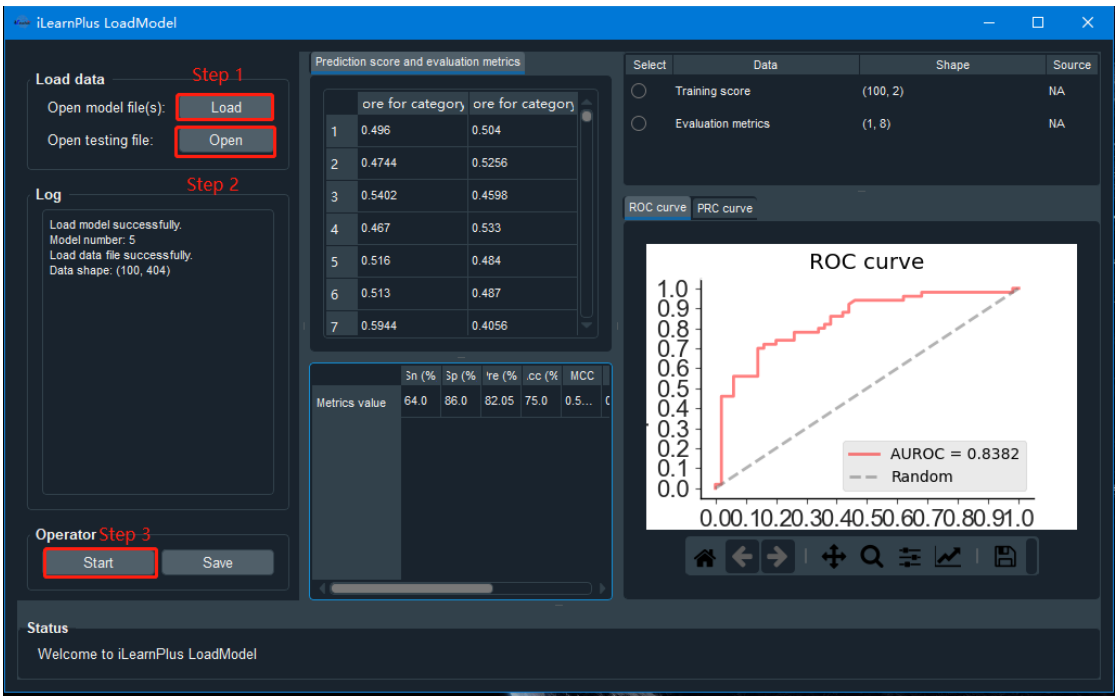


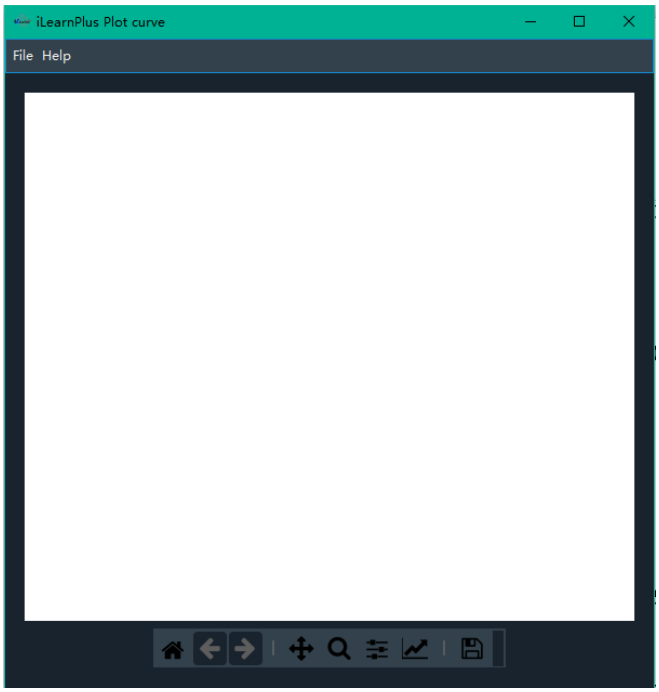
Figure S21. The interface of the *iLearnPlus-LoadModel* module.

9. Other functions

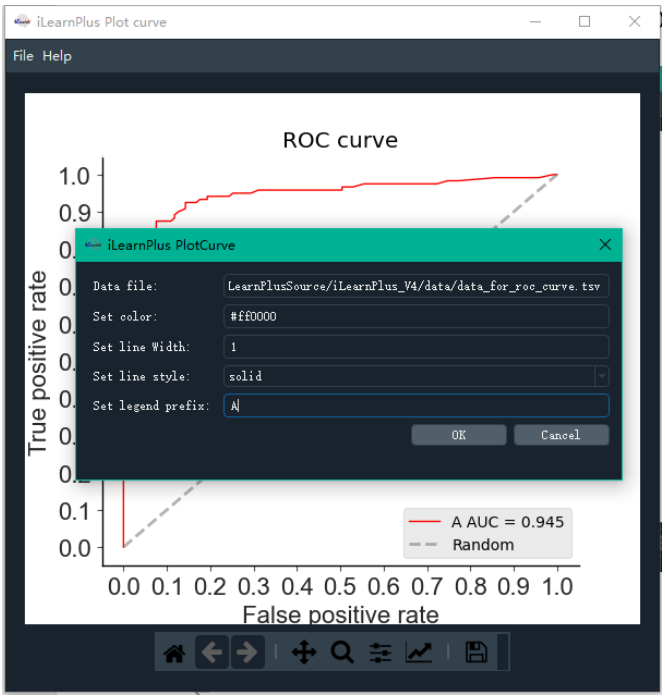
For the users' convenience, *iLearnPlus* also supplies some additional applications, including "Plot ROC curve", "Plot PRC curve", "Boxplot", "Heatmap", "Scatter plot", "Distribution visualization", "File format transformation" and "Merging feature set files". These applications aim to facilitate plotting with user-defined data and file operations.

Example of plotting the ROC curve

Step 1. Click "Visualization" → "Plot ROC curve", and open the ROC curve window.

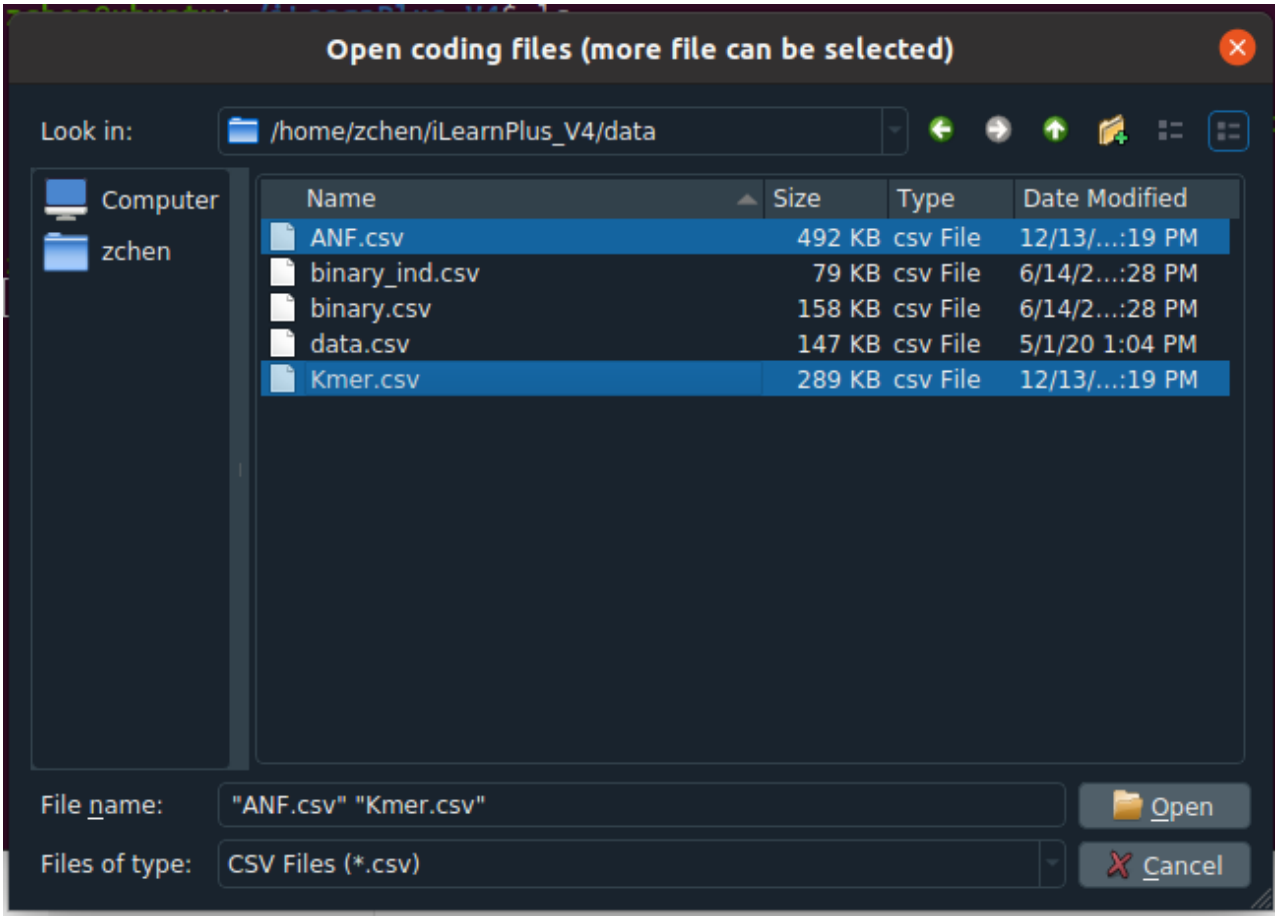


Step 2. Click "File" → "Add curve", select data file, set color and set the legend name. The example of the data file can be found in the 'data' directory of *iLearnPlus*.

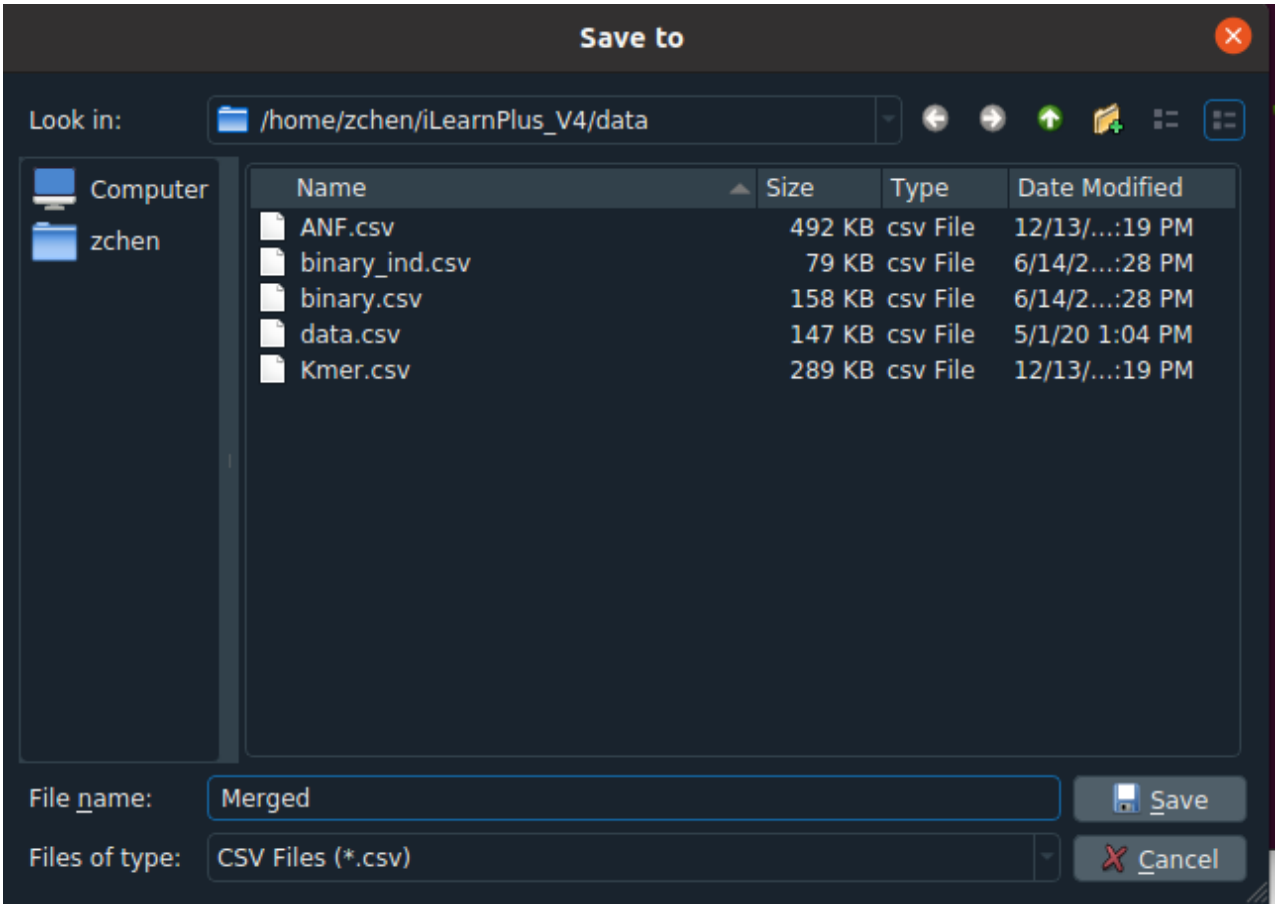


Example of merging different descriptors

Step 1. Click “Tools” → “Merge coding files into one” to open the file dialog and select the coding files that need to be merged together. At least two files should be selected by press “Ctrl” in the keyboard. Then click the ‘Open’ button.



Step 2. After clicking the ‘Open’ button, a new file dialog will be displayed. Users can select the object position of the merged coding file and set the file name of the merged coding. Then click the “Save” button to save the merged descriptors into file.



10. Performance evaluation strategy in *iLearnPlus*

As described in our paper, *iLearnPlus* supports both the binary classification task and multiclass classification task. For binary classification problems, eight frequently-used metrics are supported by *iLearnPlus*, including Sensitivity (*Sn*), Specificity (*Sp*), Accuracy (*Acc*), Matthews correlation coefficient (*MCC*), *Precision*, *F1-score*, the Area Under ROC curve (*AUROC*) and the Area Under the PRC curve (*AUPRC*). *Sn*, *Sp*, *Acc*, *MCC*, *Precision* and *F1-score* are defined as:

$$Sn = \frac{TP}{TP+FN},$$

$$Sp = \frac{TN}{TN+FP},$$

$$Acc = \frac{TP+TN}{TP+TN+FP+FN},$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}},$$

$$Precision = \frac{TP}{TP+FP} \ ,$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall},$$

where *TP*, *FP*, *TN* and *FN* represent the numbers of true positives, false positives, true negatives and false negatives, respectively. The *AUROC* and *AUCPRC* values, ranging between 0 and 1, are calculated based on the Receiver-Operating-Characteristic (ROC) curve and the Precision-Recall curve, respectively. The higher the *AUROC* and *AUPRC* values, the better the predictive performance of the model.

For multi-class classification tasks, the *Acc* is commonly used to evaluate the performance, which is defined as:

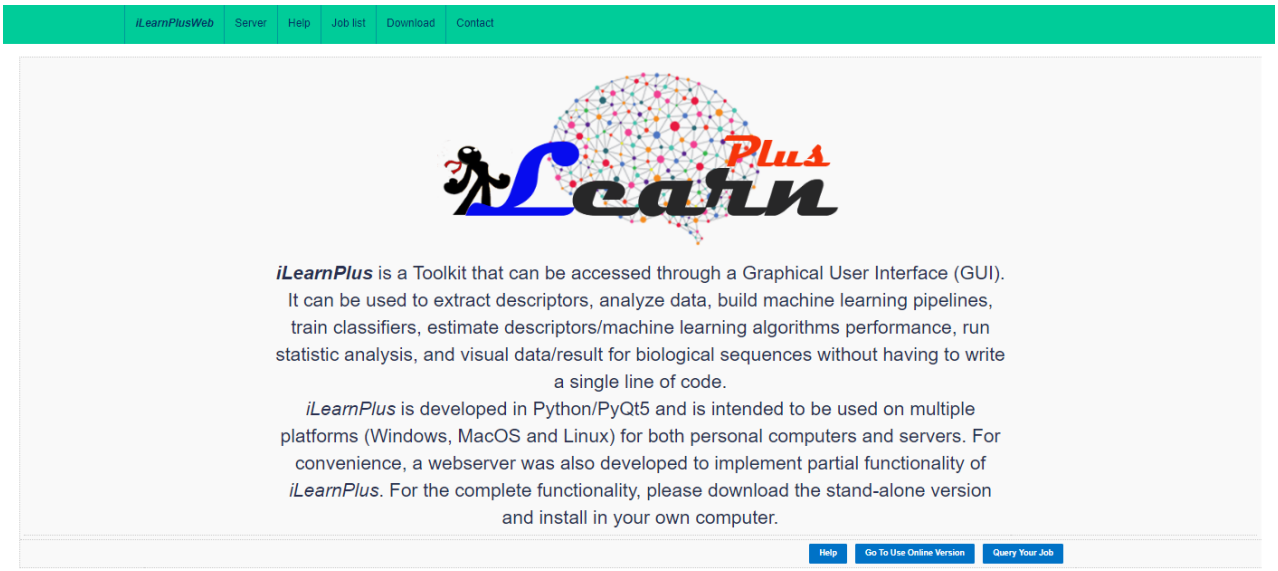
$$Acc = \frac{TP(i)+TN(i)}{TP(i)+TN(i)+FP(i)+FN(i)} \ ,$$

where *TP(i)*, *FP(i)*, *TN(i)* and *FN(i)* represent the numbers of the samples (molecules) predicted correctly to be in the *i*-th class, the total number of the samples in the *i*-th class that are predicted as one of the other classes, the total number of the samples predicted correctly not to be in the *i*-th class, and the total number of the samples not in the *i*-th class that are predicted as the *i*-th class, respectively.

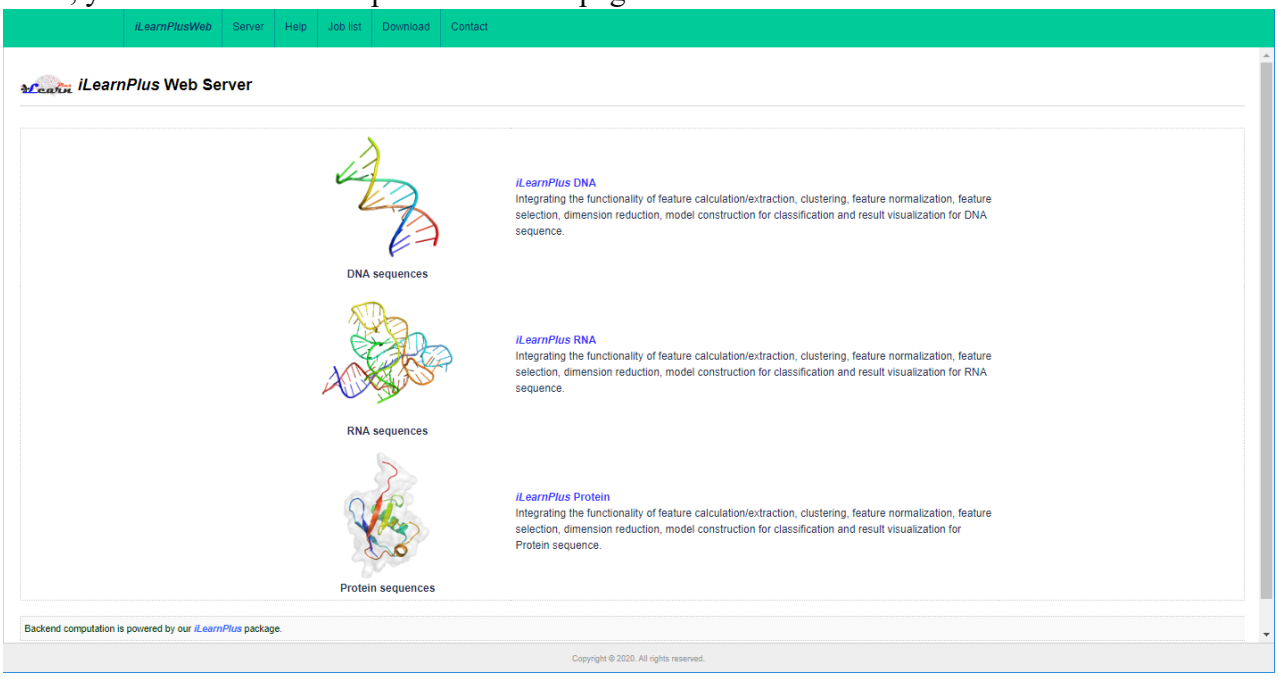
11. Online web server

The *iLearnPlus* server is freely accessible at http://ilearnplus.erc.monash.edu/, which resides on the Nectar (The National eResearch Collaboration Tools and Resources) infrastructure and managed by the eResearch Centre at Monash University. The *iLearnPlus* server was implemented based on the open-source web platform LAMP (Linux-Apache-MySQL-PHP) and is equipped with 16 cores, 64 GB memory and a 2 TB hard disk. The server has been tested across five commonly used browsers, including Internet Explorer (version ≥7.0), Microsoft Edge, Mozilla Firefox, Google Chrome and Safari. Considering the computational burden, the web server only offers the *iLearnPlus-Basic* model for the analysis and machine-learning modeling of DNA, RNA and protein sequences. The step-by-step of usage instructions is as follows:

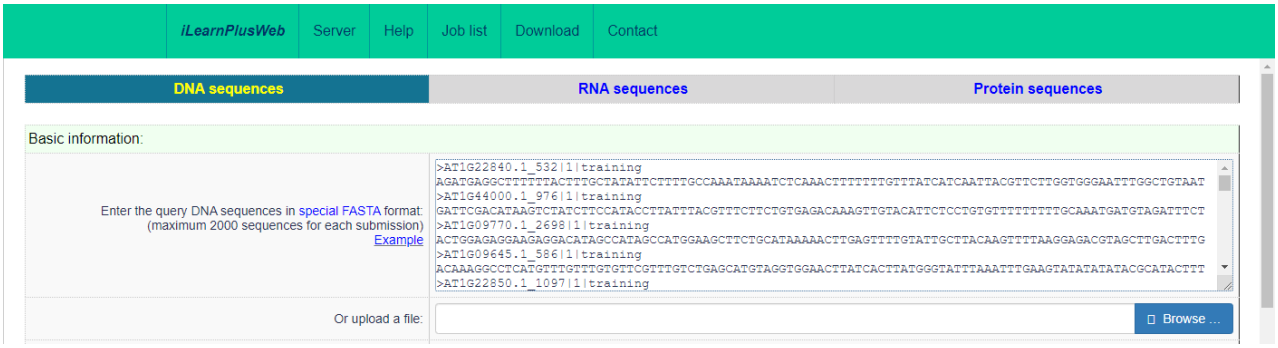
Type “http://ilearnplus.erc.monash.edu” on your browser, and click the “Go To Use Online Version” button.



Then, you will see the descriptor calculation page.

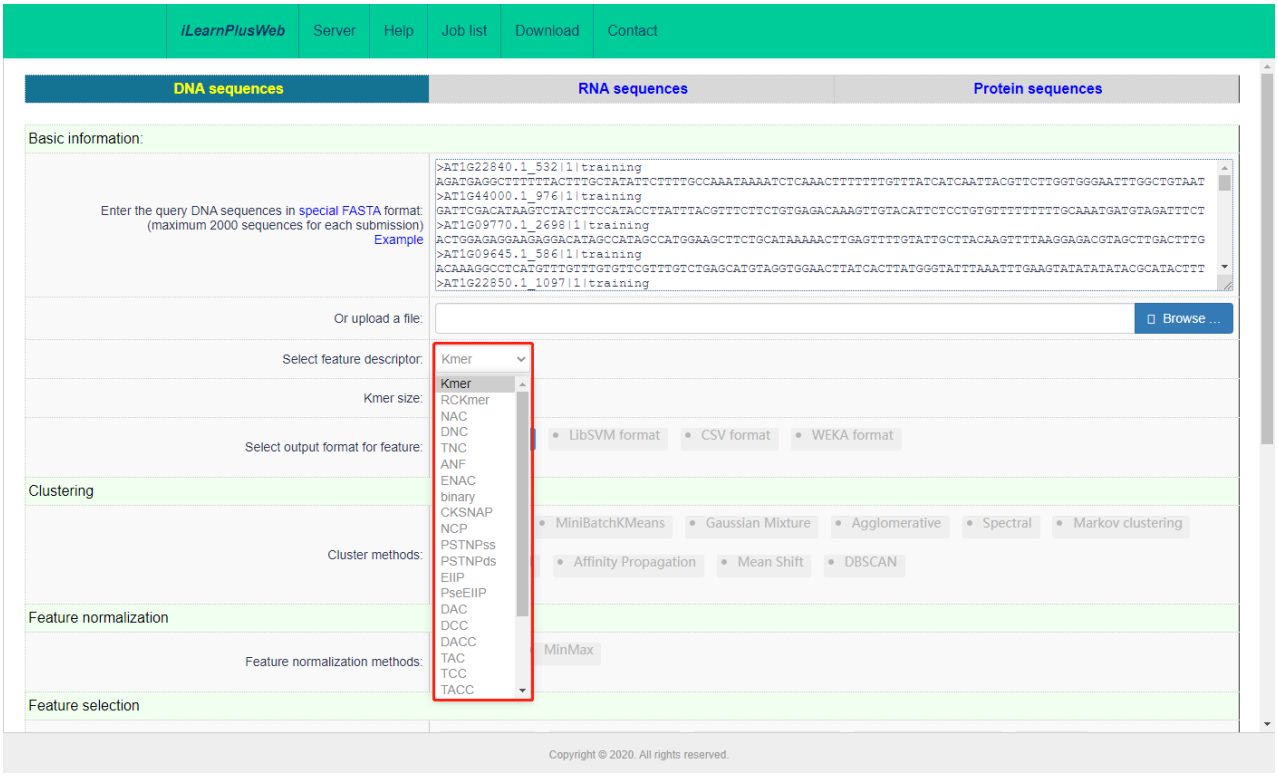


Step 1: Paste sequences or upload a sequence file.

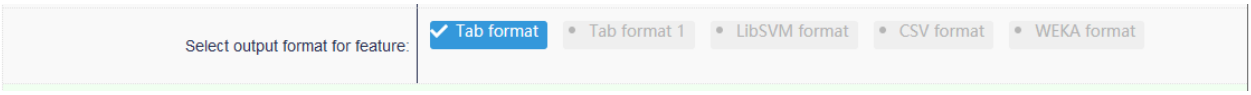


Note: Paste your protein (or peptide) sequences in the text area or upload a file that includes the sequences. The biological sequences must be in a specified 'FASTA' format. *iLearnPlus* is designed to accept no more than 2000 sequences at one time.

Step 2: Select the descriptor type.

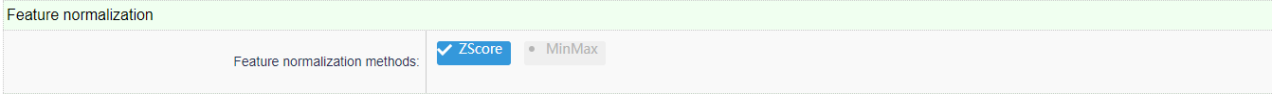


Step 3: Select the output format.

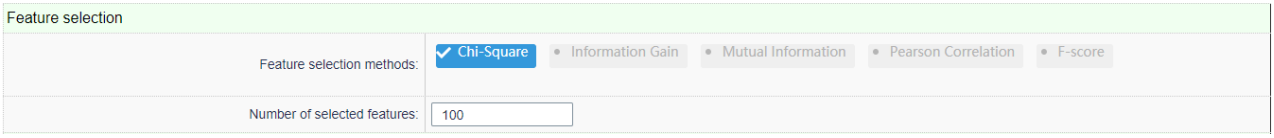


Step 4: Select the clustering method (optional).

Step 5: Select the feature normalization method (optional).

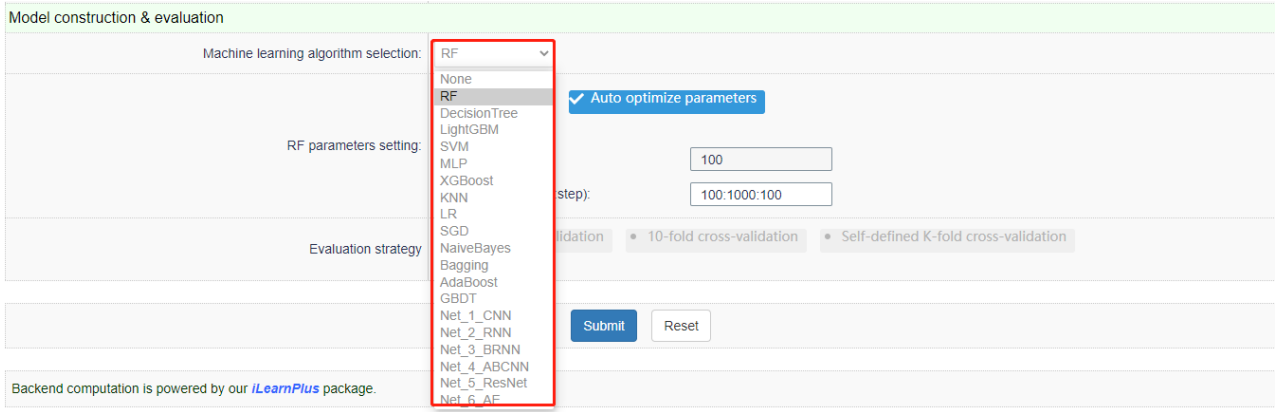


Step 6: Select the feature selection method (optional).



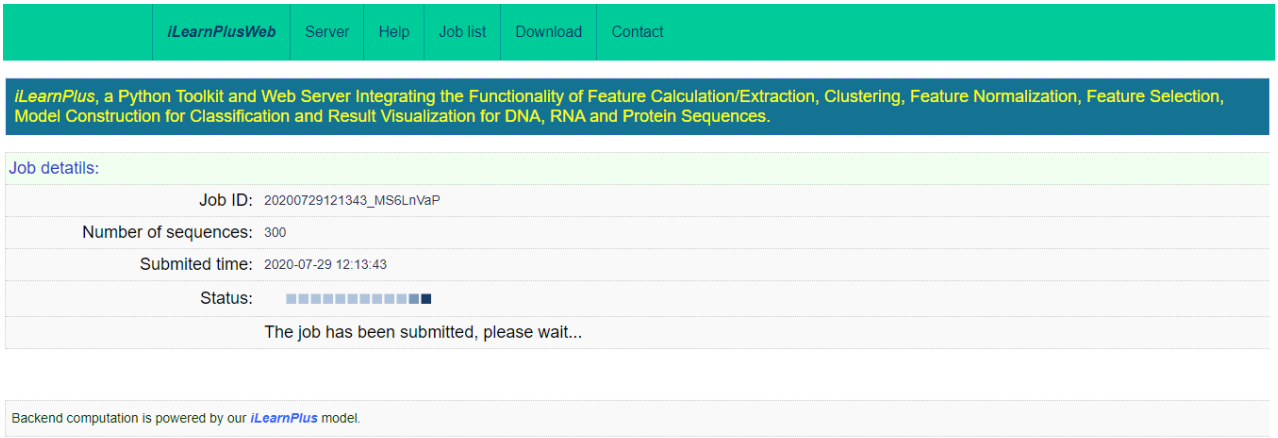
Step 7: Select the dimension reduction method (optional).

Step 8: Select the machine-learning algorithm (optional).



Finally, click the 'Submit' button to calculate the descriptors and run the selected clustering, feature selection and machine-learning algorithms.

Step 9. Wait for the prediction results.



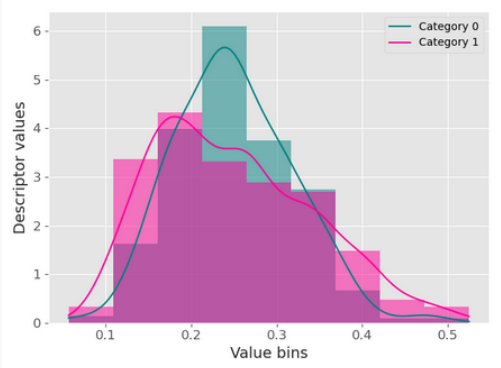
iLearnPlusWeb Job Result Details:

Job ID:	20201219230317_HxSmawD
Sequence type:	DNA
Number of training sequence:	200
Number of testing sequence:	100
Descriptor method:	NAC

Training set: View all

#	label	A	C	G	T
AT1G22840.1_532	1.0	0.2475	0.1386	0.1584	0.4554
AT1G44000.1_976	1.0	0.2277	0.1782	0.1485	0.4455
AT1G09770.1_2698	1.0	0.3069	0.1485	0.2574	0.2871
AT1G09645.1_586	1.0	0.2574	0.1287	0.2079	0.4059
AT1G22850.1_1097	1.0	0.3663	0.1089	0.2673	0.2574

Training set distribution:

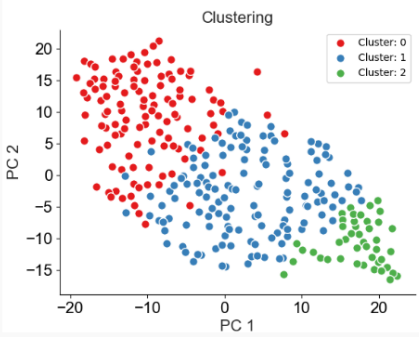


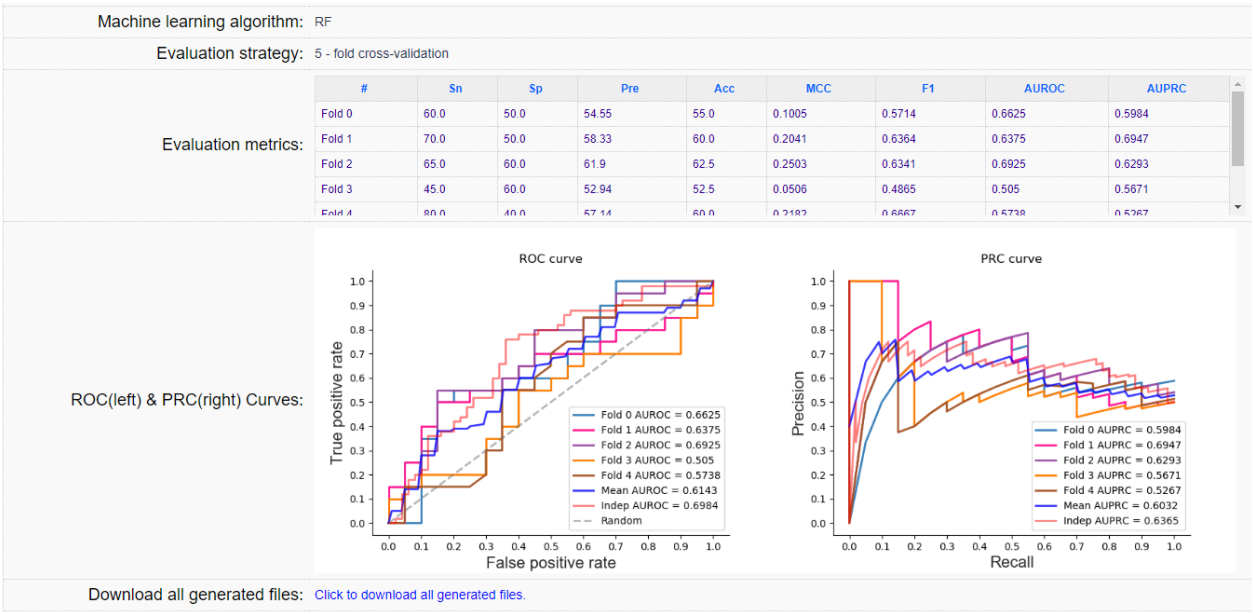
Clustering method: kmeans

Clustering result: View all

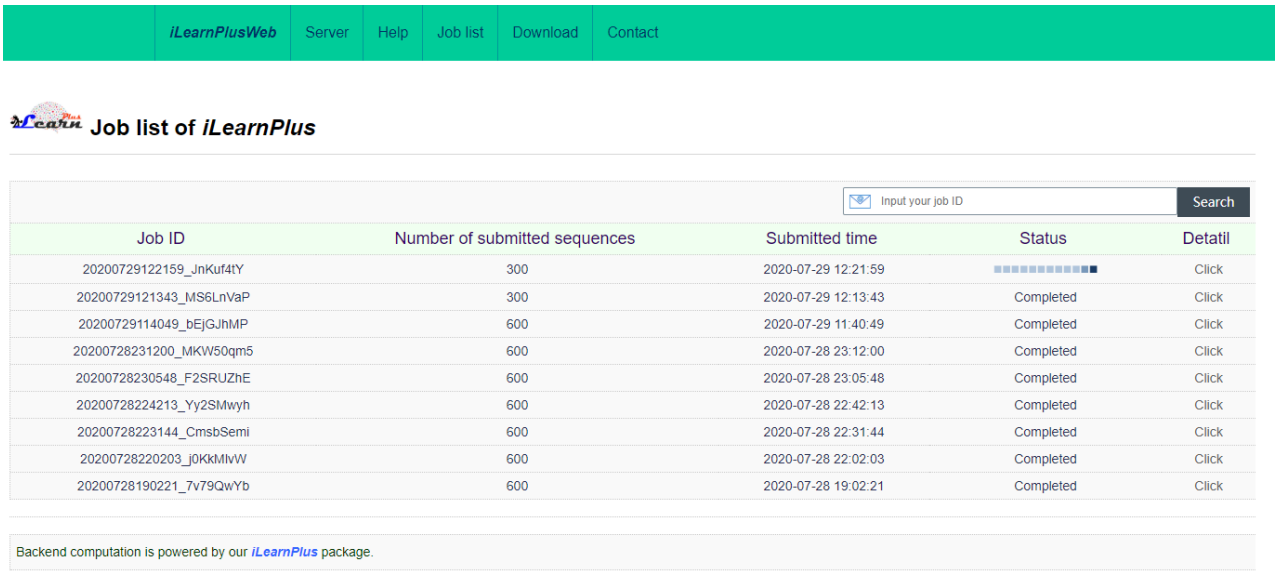
#	Cluster
AT1G22840.1_532	1
AT1G44000.1_976	2
AT1G09770.1_2698	1
AT1G09645.1_586	2
AT1G22850.1_1097	1

Visuallization of clustering result:





Step 10: Query your results.



After a few seconds, the result should display in the result page. For each job, the server will generate a job ID, and the results will be stored for a week. Within a week, you can query your result by searching your job ID.

Note: Descriptions for the feature descriptors discussed in our previous publication, iFeature (7) and iLearn (8), have been extracted from the corresponding publications.

12. Descriptions of feature descriptors for nucleotide sequences

Let us assume that a nucleotide sequence with L residues can be generally represented as $\{R_1, R_2, \ldots, R_L\}$, where R_i represents the base at the *i-th* position in the sequence. The following commonly used feature descriptors can be described as follows:

Kmer

For kmer descriptor, the DNA or RNA sequences are represented as the occurrence frequencies of k neighboring nucleic acids (9), which has been successfully applied to human gene regulatory sequence prediction (10) and enhancer identification (9). The Kmer (k=3) descriptor can be calculated as:

$$f(t) = \frac{N(t)}{N}, \quad t \in \{AAA, AAC, AAG, \ldots, TTT\},$$

where $N(t)$ is the number of kmer type t, while N is the length of a nucleotide sequence. The Kmer descriptor has been successfully applied to lncRNA prediction (11,12).

RCKmer (Reverse compliment kmer)

The reverse compliment kmer (10,13) is a variant of kmer descriptor, in which the kmers are not expected to be strand-specific. For instance, for a DNA sequence, there are 16 types of 2-mers (i.e. 'AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'CT', 'GA', 'GC', 'GG', 'GT', 'TA', 'TC', 'TG', 'TT'), 'TT' is reverse compliment with 'AA'. After removing the reverse compliment kmers, there are only 10 distinct kmers in the reverse compliment kmer approach ('AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'GA', 'GC', 'TA'). It has been used to predict the *in vivo* signature of human gene regulatory sequences (10) and human nucleosome occupancy (13).

Mismatch (The occurrence of kmers, allowing at most m mismatches)

The mismatch profile also calculates the occurrences of kmers (14), but allows max m inexact matching ($m < k$). There are two parameters for this descriptor, k neighboring nucleic acids and m inexact matching. The mismatch descriptor is defined as:

$$f_{k,m} = (\sum_{j=0}^{m} c_{1,j}, \sum_{j=0}^{m} c_{2,j}, \ldots, \sum_{j=0}^{m} c_{4^k,j}),$$

where $c_{i,j}$ represents the occurrences of i-th kmer type with j mismatches, $i = 1, 2, 3, \ldots, 4^k$; $j = 0, 1, 2, \ldots, m$. The mismatch descriptor has been successfully applied to protein classification prediction (15), B-cell epitopes identification (16) and transposon-derived piRNA prediction (17).

Subsequence (The occurrences of kmers, allowing non-contiguous matches)

The subsequence descriptor allows non-contiguous matching (14). For example, the 3-mer "AAC" in the sequence "AACTACG". By exact and non-contiguous matching, we can obtain AAC, AA-C, A-AC, A-AC ("-" means the gap in non-contiguous matching). AAC is the exact form of "AAC", and AA-C, A-AC, A-AC are non-contiguous forms of "AAC". The occurrences of non-contiguous forms are penalized with their length l and the factor δ ($0 \leq \delta \leq 1$), defined as δ^l. Therefore, the occurrence of "AAC" in above example is $1 + 2\delta^6 + \delta^5$. The subsequence descriptor has been successfully applied to B-cell epitopes identification (16), transposon-derived piRNA prediction (17).

NAC (Nucleic Acid Composition)

The Nucleic Acid Composition (NAC) encoding calculates the frequency of each nucleic acid type in a nucleotide sequence. The frequencies of all 4 natural nucleic acids (i.e. "ACGT or U") can be calculated as:

$$f(t) = \frac{N(t)}{N}, \quad t \in \{\text{A, C, G, T(U)}\},$$

where $N(t)$ is the number of nucleic acid type t, while N is the length of a nucleotide sequence.

ANF (Accumulated nucleotide frequency)

The Accumulated Nucleotide Frequency (ANF) encoding (18) include the nucleotide frequency information and the distribution of each nucleotide in the RNA sequence, the density d_i of any nucleotide s_i at position i in RNA sequence by the following formula:

$$d_i = \frac{1}{|s_i|} \sum_{j=1}^{l} f(s_i), \quad f(q) = \begin{cases} 1 & if\ s_i = q \\ 0 & other\ case \end{cases},$$

where l is the sequence length, $|S_i|$ is the length of the i-th prefix string $\{s_1, s_2, \ldots, s_i\}$ in the sequence,

$q \in$ {A, C, G or U}. Suppose an example sequence "UCGUUCAUGG". The density of 'U' is 1 (1/1), 0.5 (2/4), 0.6 (3/5), 0.5 (4/8) at positions 1, 4, 5, and 8, respectively. The density of 'C' is 0.5 (1/2), 0.33 (2/6) at positions 2 and 6, respectively. The density of 'G' is 0.33 (1/3), 0.22 (2/9), 0.3 (3/10) at positions 3, 9, and 10, respectively. The density of 'A' is 0.14 (1/7) at position 7.

By integrating both the nucleotide chemical property and accumulated nucleotide information, the sample sequence "UCGUUCAUGG" can be represented by {(0, 0, 1, 1), (0, 1, 0, 0.5), (1, 0, 0, 0.33), (0, 0, 1, 0.5), (0, 0, 1, 0.6), (0, 1, 0, 0.33), (1, 1, 1, 0.14), (0, 0, 1, 0.5), (1, 0, 0, 0.22), (1, 0, 0, 0.3)}. By doing so, not only the chemical property was considered, but also the long-range sequence order information was incorporated. Therefore, the samples in the benchmark dataset were encoded in terms of both nucleotide chemical property and nucleotide densities. The ANF descriptor has been successfully applied to N(6)-methyldenosine site prediction (18).

ENAC (Enhanced nucleic acid composition)

The Enhanced Nucleic Acid Composition (ENAC) calculates the NAC based on the sequence window of fixed length (the default value is 5) that continuously slides from the 5' to 3' terminus of each nucleotide sequence and can be usually applied to encode the nucleotide sequence with an equal length, which has been successfully applied to N(6)-methyldenosine site prediction (19).

Binary (also termed one-hot)

In the Binary encoding, each amino acid is represented by a 4-dimensional binary vector, e.g. A is encoded by (1000), C is encoded by (0100), G is encoded by (0010) and T(U) is encoded by (0001), respectively. This encoding scheme is often used to encode nucleotide sequence with an equal length. The binary descriptor has been successfully applied to N(6)-methyldenosine sites prediction (19), RNA pseudouridylation sites prediction (20), DNA- and RNA-binding proteins prediction (21) and noncoding variants effect prediction (22).

PS2 (Position-specific of two nucleotides)

There are 4 × 4 = 16 pairs of adjacent pairwise nucleotides, such as AA/AT/AG …, thus a single variable representing one such pair gets one-hot (i.e. binary) encoded into 16 binary variables (14). For example, e.g. AA is encoded by (1000000000000000), AC is encoded by

(0100000000000000) …, and the sequence AAC is encoded as (1000000000000000010000000000000000). PS2 has been successfully applied to off-target effects of CRISPR-Cas9 prediction (23). Both PS3 and PS4 are encoded for three adjacent nucleotides (4 × 4 × 4 = 64) and four adjacent nucleotides (4 × 4 × 4 × 4 = 256) in a similar way.

CKSNAP (Composition of *k*-spaced nucleic acid pairs)

The CKSNAP feature encoding calculates the frequency of nucleic acid pairs separated by any *k* nucleic acid (*k* = 0, 1, 2, … , 5) (8). Taking *k* = 0 as an example, there are 16 0-spaced nucleic acid pairs (i.e. 'AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'CT', 'GA', 'GC', 'GG', 'GT', 'TA', 'TC', 'TG', 'TT'). Then, a feature vector can be defined as:

$$(\frac{N_{AA}}{N_{total}}, \frac{N_{Ac}}{N_{total}}, \frac{N_{AG}}{N_{total}}, \dots, \frac{N_{TT}}{N_{total}},)_{16}.$$

The value of each descriptor denotes the composition of the corresponding nucleic acid pair in the nucleotide sequence. For instance, if the nucleic acid pair AA appears *m* times in the nucleotide sequence, the composition of the nucleic acid pair AA is equal to *m* divided by the total number of *0*-spaced nucleic acid pairs (N_{total}) in the nucleotide sequence. For *k* = 0, 1, 2, 3, 4 and 5, the value of N_{total} is $P-1$, $P-2$, $P-3$, $P-4$, $P-5$ and $P-6$ for a nucleotide sequence of length *P*, respectively.

NCP (Nucleotide chemical property)

There are four different kinds of nucleotides in RNA, i.e., adenine (A), guanine (G), cytosine (C) and uracil (U). Each nucleotide has different chemical structure and chemical binding. The four kinds of nucleotides can be classified into three different groups in terms of these chemical properties (**Table S2**).

Table S2. Chemical structure of each nucleotide (18).

Chemical property	Class	Nucleotides
Ring Structure	Purine	A, G
	Pyrimidine	C, U
Functional Group	Amino	A, C
	Keto	G, U
Hydrogen Bond	Strong	C, G
	Weak	A, U

Based on chemical properties, A can be represented by coordinates (1, 1, 1), C can be represented

by coordinates (0, 1, 0), G can be represented by coordinates (1, 0, 0), U can be represented by coordinates (0, 0, 1). The NCP descriptor has been successfully applied to N(6)-methyldenosine site prediction (18).

PSTNPss (Position-specific trinucleotide propensity based on single-strand)

The Position-specific trinucleotide propensity based on single-strand (PSTNPss) (24,25) using a statistical strategy based on single-stranded characteristics of DNA or RNA. There are $4^3 = 64$ trinucleotides: AAA, AAC, AAG, ..., TTT(UUU). So, for an L bp sample, its details of the trinucleotides position specificity can be expressed by the following 64 × (L-2) matrix:

$$\mathrm{Z} = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,L-2} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,L-2} \\ \vdots & \vdots & \cdots & \vdots \\ z_{64,1} & z_{64,2} & \cdots & z_{64,L-2} \end{bmatrix}$$

where

$$z_{i,j} = F^{+}(3mer_i|j) - F^{-}(3mer_i|j), i = 1,2,\ldots,64; j = 1,2,\ldots L-2$$

$F^{+}(3mer_i|j)$ and $F^{-}(3mer_i|j)$ denote the frequency of the i-th trinucleotide ($3mer_i$) at the j-th position appear in the positive (S^{+}) and negative (S^{-}) data sets, respectively. In the formula, $3mer_1$ equals AAA,$3mer_2$ equals AAC, …, $3mer_{64}$ equals TTT. Therefore, the sample can be expressed as:

$$\mathrm{S} = \left[\emptyset_1, \emptyset_2, \ldots, \emptyset_{L-2}\right]^T$$

where T is the operator of transpose and ϕ_u was defined as follows:

$$\emptyset_u = \begin{cases} z_{1,u}, & when\ N_u N_{u+1} N_{u+2} = AAA \\ z_{2,u}, & when\ N_u N_{u+1} N_{u+2} = AAG \\ \vdots & \\ z_{64,u}, & when\ N_u N_{u+1} N_{u+2} = TTT \end{cases}$$

The PSTNP descriptor has been successfully applied to DNA N4-methylcytosine sites prediction (25).

PSTNPds (Position-specific trinucleotide propensity based on double-strand)

Feature Position-specific trinucleotide propensity based on double-strand (PSTNPds) (24,25) using a statistical strategy based on double-stranded characteristics of DNA according to complementary base pairing, so they have more evident statistical features. At this point, we deem A and T as

identical, the same to C and G. Thus, for every sample, it can be converted into a sequence contained A and T only.

EIIP (Electron-ion interaction pseudopotentials)

Dragutin (26) came up with electron-ion interaction pseudopotentials (EIIP) value of nucleotides A, G, C, T (A: 0.1260, C: 0.1340, G: 0.0806, T:0.1335). The EIIP directly use the EIIP value represent the nucleotide in the DNA sequence. Therefore, the dimension of the EIIP descriptor is the length of the DNA sequence. The EIIP descriptor has been successfully applied to DNA N4-methylcytosine site prediction (27) and subcellular location prediction (28).

PseEIIP (Electron-ion interaction pseudopotentials of trinucleotide)

In these encoding, let $EIIP_A$, $EIIP_T$, $EIIP_G$, and $EIIP_C$ denote the EIIP values of nucleotides A, T, G and C, respectively. Then, the mean EIIP value of trinucleotides in each sample to construct feature vector, which can be formulated as:

$$\mathrm{V} = [EIIP_{AAA} \cdot f_{AAA}, EIIP_{AAC} \cdot f_{AAC}, \dots, EIIP_{TTT} \cdot f_{TTT}],$$

where f_{xyz} is the normalized frequency of the i-th trinucleotide, $EIIP_{xyz} = EIIP_x + EIIP_y + EIIP_z$ expresses the *EIIP* value of one trinucleotide and X, Y, Z∈ [A, C, G, T]. Obviously, the dimension of vector V is 64. The EIIP descriptor has been successfully applied to 5-Methylcytosine prediction (29) and non-coding RNA promoter identification (30).

ASDC (Adaptive skip dinucleotide composition)

The adaptive skip dipeptide composition is a modified dinucleotide composition, which sufficiently considers the correlation information present not only between adjacent residues but also between intervening residues (31). For given a sequence, the feature vector for ASDC is represented by:

$$ASDC = (f_{v1}, f_{v1}, \dots, f_{v16}),$$

where f_{vi} is calculated by

$$f_{vi} = \frac{\sum_{g=1}^{L-1} O_i^g}{\sum_{i=1}^{16} \sum_{g=1}^{L-1} O_i^g},$$

where f_{vi} denotes the occurrence frequency of all possible dinucleotide with ≤ L-1 intervening nucleotides. The ASDC descriptor has been successfully applied to anti-cancer peptide (31) and cell-

penetrating peptide predictions (32).

DBE (Dinucleotide binary encoding)

The dinucleotide binary encoding descriptor encapsulates the positional information of the dinucleotide at each position in the sequence. There are a total of 16 possible dinucleotides. In this descriptor, each dinucleotide can be encoded into a 4-dimensional 0/1 vector. For example, AA is encoded as (0,0,0,0); AT is encoded as (0,0,0,1); AC is encoded as (0,0,1,0); and so forth, GG is encoded as (1,1,1,1). Therefore, using the dinucleotide binary encoding, we can yield a 160 (=40×4)-dimensional 0/1 vector for the given sequence. The DBE descriptor has been successfully applied to N6-Methyladenosine site prediction (33).

LPDF (Local position-specific dinucleotide frequency)

The local position-specific dinucleotide frequency descriptor can be denoted as ($f_2, f_3, \ldots, f_l$), where f_i is calculated as follows:

$$f = \frac{1}{|N_i|} C(X_{i-1}X_i), 2 \le i \le l,$$

where l is the length of the given sequence, $|N_i|$ is the length of the i-th prefix string $\{X_1X_2 \ldots X_i\}$ in the sequence, and C ($X_{i-1}X_i$) is the occurrence number of the dinucleotide $X_{i-1}X_i$ in position i of the i-th prefix string. The LPDF descriptor has been successfully applied to N6-Methyladenosine site prediction (33).

DPCP (Dinucleotide physicochemical properties)

The dinucleotide physicochemical properties descriptor can be devised as:

$$V = [DPCP_{AA} \times f_{AA}, DPCP_{AC} \times f_{Ac}, \ldots, DPCP_{TT} \times f_{TT}],$$

where $DPCP_i$ is one of the i-th physicochemical properties of a dinucleotide. The 148 physicochemical properties for DNA dinucleotide and 22 physicochemical properties for RNA dinucleotide are listed in **Tables S3** and **S4**, respectively. The DPCP descriptor has been successfully applied to DNA N4-Methylcytosine sites prediction (34).

Table S3. The names of the 148 physicochemical dinucleotides indices for DNA.

Base stacking	Protein induced deformability	B-DNA twist	Propeller twist	Duplex stability:(freeenergy)
Duplex tability(disruptenergy)	Protein DNA twist	Stabilising energy of Z-DNA	Aida_BA_transition	Breslauer_dS
Electron_interaction	Hartman_trans_free_energy	Lisser_BZ_transition	Polar_interaction	SantaLucia_dG
Sarai_flexibility	Stability	Stacking_energy	Sugimoto_dS	Watson-Crick_interaction
Twist	Shift	Slide	Rise	Twist stiffness
Tilt stiffness	Shift_rise	Twist_shift	Enthalpy1	Twist_twist
Shift2	Tilt3	Tilt1	Slide (DNA-protein complex) 1	Tilt_shift
Twist_tilt	Roll_rise	Stacking energy	Stacking energy1	Propeller Twist
Roll11	Rise (DNA-protein complex)	Roll2	Roll3	Roll1
Slide_slide	Enthalpy	Shift_shift	Flexibility_slide	Minor Groove Distance
Rise (DNA-protein complex)1	Roll (DNA-protein complex)1	Entropy	Cytosine content	Major Groove Distance
Twist (DNA-protein complex)	Purine (AG) content	Tilt_slide	Major Groove Width	Major Groove Depth
Free energy6	Free energy7	Free energy4	Free energy3	Free energy1
Twist_roll	Flexibility_shift	Shift (DNA-protein complex) 1	Thymine content	Tip
Keto (GT) content	Roll stiffness	Entropy1	Roll_slide	Slide (DNA-protein complex)
Twist2	Twist5	Twist4	Tilt (DNA-protein complex)1	Twist_slide
Minor Groove Depth	Persistance Length	Rise3	Shift stiffness	Slide3
Slide2	Slide1	Rise1	Rise stiffness	Mobility to bend towards minor groove
Dinucleotide GC Content	A-philicity	Wedge	DNA denaturation	Bending stiffness
Free energy5	Breslauer_dG	Breslauer_dH	Shift (DNA-protein complex)	Helix-Coil_transition
Ivanov_BA_transition	Slide_rise	SantaLucia_dH	SantaLucia_dS	Minor Groove Width
Sugimoto_dG	Sugimoto_dH	Twist1	Tilt	Roll
Twist7	Clash Strength	Roll_roll	Roll (DNA-protein complex)	Adenine content
Direction	Probability contacting nucleosome core	Roll_shift	Shift_slide	Shift1
Tilt4	Tilt2	Free energy8	Twist (DNA-protein complex)1	Tilt_rise
Free energy2	Stacking energy2	Stacking energy3	Rise_rise	Tilt_tilt
Roll4	Tilt_roll	Minor Groove Size	GC content	Inclination
Slide stiffness	Melting Temperature1	Twist3	Tilt (DNA-protein complex)	Guanine content
Twist6	Major Groove Size	Twist_rise	Rise2	Melting Temperature
Free energy	Mobility to bend towards major groove	Bend		

Table S4. The names of the 22 physicochemical dinucleotides indices for RNA.

Shift (RNA)	Hydrophilicity (RNA)	Hydrophilicity (RNA)	GC content	Purine (AG) content
Keto (GT) content	Adenine content	Guanine content	Cytosine content	Thymine content
Slide (RNA)	Rise (RNA)	Tilt (RNA)	Roll (RNA)	Twist (RNA)
Stacking energy (RNA)	Enthalpy (RNA)	Entropy (RNA)	Free energy (RNA)	Free energy (RNA)
Enthalpy (RNA)	Entropy (RNA)			

DPCP type2 (Dinucleotide physicochemical properties)

For DPCP type2 descriptor, it can be encoded into the following matrix:

$$V = \begin{bmatrix} PC^1(N_1N_2) & \dots & PC^1(N_{L-1}N_L) \\ \dots & \dots & \dots \\ PC^j(N_1N_2) & \dots & PC^j(N_{L-1}N_L) \end{bmatrix},$$

where $PC_j(N_i, N_{i+1})$ is the j-th physicochemical dinucleotides value of the dinucleotide N_iN_{i+1}. L is the length of sequence. The 148 physicochemical properties for DNA dinucleotide and 22 physicochemical properties for RNA dinucleotide are listed in **Tables S3** and **S4**, respectively. The DPCP type2 descriptor has been successfully applied to DNA N4-Methylcytosine sites prediction (35).

TPCP (Trinucleotide physicochemical properties)

The trinucleotide physicochemical properties descriptor can be devised as:

$$V = [TPCP_{AAA} \times f_{AAA}, TPCP_{AAC} \times f_{AAC}, \dots, DPCP_{TTT} \times f_{TTT}],$$

where $TPCP_i$ is one of the i-th physicochemical properties of a trinucleotide listed in **Table S4**, and f_{NNN} denotes the trinucleotide normalized frequency. The 12 physicochemical properties for DNA trinucleotide are listed in **Table S5**. The TPCP descriptor has been successfully applied to DNA N4-Methylcytosine site prediction (34).

Table S5. The names of the 12 physicochemical trinucleotides indices for DNA.

Dnase I	Bendability (DNAse)	Bendability (consensus)	Trinucleotide GC Content
Nucleosome positioning	Consensus_roll	Consensus-Rigid	Dnase I-Rigid
MW-Daltons	MW-kg	Nucleosome	Nucleosome-Rigid

TPCP type2 (Trinucleotide physicochemical properties)

For TPCP type2 descriptor, it can be encoded into the following matrix:

$$V = \begin{bmatrix} PC^1(N_1N_2N_3) & \dots & PC^1(N_{L-2}N_{L-1}N_L) \\ \dots & \dots & \dots \\ PC^j(N_1N_2N_3) & \dots & PC^j(N_{L-2}N_{L-1}N_L) \end{bmatrix}$$

Where $PC_j(N_i, N_{i+1},N_{i+2})$ is the j-th physicochemical value of the trinucleotide N_iN_{i+1},N_{i+2}. L is the length of sequence. The 12 physicochemical properties for DNA trinucleotide are listed in **Table S5**. The TPCP type2 descriptor has been successfully applied to DNA N4-Methylcytosine site prediction (35).

MMI (Multivariate mutual information)

In order to use multivariate mutual information on a DNA/RNA sequence, the '2-mer' set T2 = { AA,

AC, AG, AT, CC, CG, CT, GG, GT, TT} and '3-mer' set T3 = {AAA, AAC, AAG, AAT, ACC, ACG, ACT, AGG, AGT, ATT, CCC, CCG, CCT, CGG, CGT, CTT, GGG, GGT, GTT and TTT} were defined, and then the mutual information can be calculated as follows:

$$I(N_1N_2) = f(N_1N_2)ln\frac{f(N_1N_2)}{f(N_1)f(N_2)}$$

$$I(N_1N_2N_3) = f(N_1N_2)ln\frac{f(N_1N_2)}{f(N_1)f(N_2)} + \frac{f(N_1N_3)}{f(N_3)}ln\frac{f(N_1N_3)}{f(N_3)} - \frac{f(N_1N_2N_3)}{f(N_2N_3)}ln\frac{f(N_1N_2N_3)}{f(N_2N_3)}$$

where $f(N_i)$ is the frequency of N_i in the sequence; $f(N_i, N_j)$ is the frequency of the T2's element N_iN_j; $f(N_i, N_j, N_k)$ are the frequency of the T3's element $N_iN_jN_k$. The MMI descriptor has been successfully applied to DNA N4-Methylcytosine site prediction (35,36).

KNN (K-nearest neighbor)

The K-nearest neighbor descriptor depicts how much one query sample resembles other samples. When comparing two local sequence fragments A and B, the similarity score $S(A, B)$ can be calculated as:

$$S(A,B) = \sum_{1\le i\le L} Score(A_i, B_i),$$

where L is sequence length; A_i is the nucleotide at position i in the sequence A; B_i is the nucleotide at position i in the sequence B. The similarity score for A and B is given by:

$$Score = \begin{cases} 2, if\ a = b \\ -1, others \end{cases}.$$

The KNN descriptor has been successfully applied to DNA N4-Methylcytosine sites prediction (35) and RNA N6-methyladenosine site prediction (37).

Z_curve_9bit (The Z curve parameters for frequencies of phase-specific mononucleotides)

The frequencies of bases A, C, G and T occurring in an open reading frame or a fragment of DNA sequence with base at position 1, 4, 7, …; 2, 5, 8, …; 3, 6, 9, …, are denoted by a_1, c_1, g_1, t_1; a_2, c_2, g_2, t_2; a_3, c_3, g_3, t_3, respectively. They are in fact the frequencies of bases at the 1st, 2nd and 3rd codon positions. a_i, c_i, g_i, t_i are mapped onto a point P_i in a three-dimensional space V_i, i=1, 2, 3. The coordinates of P_i, denoted by x_i, y_i, z_i, are determined by the Z-transform of DNA sequences:

$$\begin{cases} x_i = (a_i + g_i) - (c_i + t_i), \\ y_i = (a_i + c_i) - (g_i + t_i), \\ z_i = (a_i + t_i) - (g_i + c_i), \\ x_i, y_i, z_i, \in [-1, 1], i = 1,2,3. \end{cases}$$

Z_curve_12bit (The Z curve parameters for frequencies of phaseindependent di-nucleotides)

The Z_curve_12bit descriptor consider the frequency of dinucleotides, denoted by *p(XY)*, where X, Y = A, C, G and T. This descriptor can be calculated as follows:

$$\begin{cases} x_X = (p(XA) + p(XG)) - (p(XC) + p(XT)), \\ y_X = (p(XA) + p(XC)) - (p(XG) + p(XT)), \\ z_X = (p(XA) + p(XT)) - (p(XG) + p(XC)), \\ X = A, C, G, T, \end{cases}$$

Z_curve_36bit (The Z curve parameters for frequencies of phase-specific di-nucleotides)

Using similar notations, the Z_curve_36bit descriptor can be calculated as follows:

$$\begin{cases} x_X^k = (p^k(XA) + p^k(XG)) - (p^k(XC) + p^k(XT)), \\ y_X^k = (p^k(XA) + p^k(XC)) - (p^k(XG) + p^k(XT)), \\ z_X^k = (p^k(XA) + p^k(XT)) - (p^k(XG) + p^k(XC)), \\ X = A, C, G, T; k = 1,2,3, \end{cases}$$

Z_curve_48bit (The Z curve parameters for frequencies of phaseindependent tri-nucleotides)

Using similar notations, the Z_curve_48bit descriptor can be calculated as follows:

$$\begin{cases} x_{XY} = (p(XYA) + p(XYG)) - (p(XYC) + p(XYT)), \\ y_{XY} = (p(XYA) + p(XYC)) - (p(XYG) + p(XYT)), \\ z_{XY} = (p(XYA) + p(XYT)) - (p(XYG) + p(XYC)), \\ X = A, C, G, T; Y = A, C, G, T \end{cases}$$

Z_curve_144bit (The Z curve parameters for frequencies of phase-specific tri-nucleotides)

Using similar notations, the Z_curve_144bit descriptor can be calculated as follows:

$$\begin{cases} x_{XY}^k = (p^k(XYA) + p^k(XYG)) - (p^k(XYC) + p^k(XYT)), \\ y_{XY}^k = (p^k(XYA) + p^k(XYC)) - (p^k(XYG) + p^k(XYT)), \\ z_{XY}^k = (p^k(XYA) + p^k(XYT)) - (p^k(XYG) + p^k(XYC)), \\ X = A, C, G, T; Y = A, C, G, T; k = 1,2,3. \end{cases}$$

The Z_curve descriptor has been successfully applied to short coding sequence identification of human genes (38).

Autocorrelation

The Autocorrelation encoding (39) can transform the nucleotide/protein sequences of different lengths into fixed-length vectors by measuring the correlation between any two properties. Autocorrelation encoding can generate two kinds of variables (i.e. The autocorrelation (AC) between the same property, and the cross-covariance (CC) between two different properties). There are six types of autocorrelation encodings for nucleotide sequence, including dinucleotide-based auto covariance (DAC), dinucleotide-based cross covariance (DCC), dinucleotide-based auto-cross covariance (DACC), trinucleotide-based auto covariance (TAC), trinucleotide-based cross covariance (TCC), and trinucleotide-based auto-cross covariance (TACC). The used 148 physicochemical properties for DNA dinucleotide and 22 physicochemical properties for RNA dinucleotide are listed in **Tables S3** and **S4**, respectively. The used 12 physicochemical properties for DNA trinucleotide are listed in **Table S5**.

DAC (Dinucleotide-based auto covariance)

The Dinucleotide-based Auto Covariance (DAC) encoding (39) measures the correlation of the same physicochemical index between two dinucleotide separated by a distance of *lag* along the sequence. The DAC can be calculated as:

$$DAC(u, lag) = \sum_{i=1}^{L-lag-1} ((P_u(R_iR_{i+1}) - \overline{P_u})(P_u\left(R_{i+lag}R_{i+lag+1}\right) - \overline{P_u})/(L - lag - 1)),$$

where u is a physicochemical index, L is the length of the nucleotide sequence, $P_u(R_iR_{i+1})$ is the numerical value of the physicochemical index u for the dinucleotide R_iR_{i+1} at position i, $\overline{P_u}$ is the average value for physicochemical index u along the whole sequence:

$$\overline{P_u} = \sum_{j=1}^{L-1} P_u\left(R_jR_{j+1}\right)/(L-1).$$

The dimension of the DAC feature vector is $N \times$ LAG, where N is the number of physicochemical indices and LAG is the maximum of *lag* (*lag* = 1, 2, …, LAG).

DCC (Dinucleotide-based Cross Covariance)

The Dinucleotide-based Cross Covariance (DCC) encoding (39) measures the correlation of two different physicochemical indices between two dinucleotides separated by *lag* nucleic acids along the sequence. The DCC encoding is calculated as:

$$\mathrm{DCC}(u_1, u_2, lag) = \sum_{i=1}^{L-lag-1} \left(P_{u_1}(R_i R_{i+1}) - \bar{P}_{u_1}\right)\left(P_{u_2}\left(R_{i+lag} R_{i+lag+1}\right) - \bar{P}_{u_2}\right)/(L - lag - 1),$$

where u_1 and u_2 are different physicochemical indices, L is the length of the nucleotide sequence, $P_{u_a}(R_i R_{i+1})$ is the numerical value of the physicochemical index u_a for the dinucleotide $R_i R_{i+1}$ at position i, $\bar{P}_{u_a}$ is the average value for physicochemical index u_a along the whole sequence:

$$\bar{P}_{u_a} = \sum_{j=1}^{L-1} P_{u_a}\left(R_j R_{j+1}\right)/(L-1).$$

The dimension of the DCC feature vector is $N \times (N\text{-}1) \times \mathrm{LAG}$, where N is the number of physicochemical indices and LAG is the maximum of *lag* (*lag* = 1, 2, …, LAG).

DACC (Dinucleotide-based Auto-Cross Covariance)

The Dinucleotide-based Auto-Cross Covariance (DACC) encoding (39) is a combination of DAC and DCC encoding. Thus, the dimension of the DACC encoding is $N \times N \times \mathrm{LAG}$, where N is the number of physicochemical indices and LAG is the maximum of the *lag* (*lag* = 1, 2, …, LAG).

TAC (Trinucleotide-based Auto Covariance)

The Trinucleotide-based Auto Covariance (TAC) encoding measures the correlation of the same physicochemical index between trinucleotides separated by lag nucleic acids along the sequence, and can be calculated as:

$$TAC(lag, u) = \sum_{i=1}^{L-lag-2} (P_u(R_i R_{i+1} R_{i+2}) - \bar{P}_u)(P_u\left(R_{i+lag} R_{i+lag+1} R_{i+lag+2}\right) - \bar{P}_u)/(L - lag - 2),$$

where u is a physicochemical index, L is the length of the nucleotide sequence, $P_u(R_i R_{i+1} R_{i+2})$ is the numerical value of the physicochemical index u for the trinucleotide $R_i R_{i+1} R_{i+2}$ at position i, $\bar{P}_u$ is the average value for physicochemical index u along the whole sequence:

$$\bar{P}_u = \sum_{j=1}^{L-2} P_u(R_i R_{i+1} R_{i+2}) / (L-2).$$

The dimension of the TAC feature vector is $N \times$ LAG, where N is the number of physicochemical indices and LAG is the maximum of *lag* (*lag* = 1, 2, …, LAG).

TCC (Trinucleotide-based Cross Covariance)

The Trinucleotide-based Cross Covariance (TCC) encoding measures the correlation of two different physicochemical indices between two trinucleotides separated by *lag* nucleic acids along the sequence. The TCC encoding can be calculated as:

$$\mathrm{DCC}(u_1, u_2, lag) = \sum_{i=1}^{L-lag-2} \left(P_{u_1}(R_i R_{i+1} R_{i+2}) - \bar{P}_{u_1}\right)\left(P_{u_2}\left(R_{i+lag} R_{i+lag+1} R_{i+lag+2}\right) - \bar{P}_{u_2}\right)/(L - lag - 2)$$

where u_1 and u_2 are different physicochemical indices, L is the length of the nucleotide sequence, $R_i R_{i+1} R_{i+2}$ is the numerical value of the physicochemical index u_a for the dinucleotide $R_i R_{i+1} R_{i+2}$ at position i, $\bar{P}_{u_a}$ is the average value for physicochemical index u_a along the whole sequence:

$$\bar{P}_{u_a} = \sum_{j=1}^{L-1} P_{u_a}\left(R_j R_{j+1} R_{j+2}\right) / (L-2)$$

The dimension of the DCC feature vector is $N \times (N\text{-}1) \times$ LAG, where N is the number of physicochemical indices and LAG is the maximum of *lag* (*lag* = 1, 2, …, LAG).

TACC (Trinucleotide-based Auto-Cross Covariance)

Like DAC encoding, the Trinucleotide-based Auto-Cross Covariance (TACC) encoding (39) is a combination of TAC and TACC encoding. Thus, the dimension of the TACC encoding is $N \times N \times$ LAG, where N is the number of physicochemical indices and LAG is the maximum of the *lag* (*lag* = 1, 2, …, LAG). There are three types of autocorrelation encodings for protein sequence, including auto covariance (AC), cross covariance (CC), auto-cross covariance (ACC), The amino acid properties used here are different types of amino acids index, which is retrieved from the AAindex Database (40) available at http://www.genome.jp/dbget/aaindex.html. The Autocorrelation descriptor has been successfully applied to protein fold recognition (41) and protein-protein interaction prediction (42).

PseNAC (Pseudo nucleic acid composition)

The Pseudo Nucleic Acid Composition (PseNAC) encodings consider both the local sequence-order information and long-range sequence-order effects (39,43). Six types of PseNAC encodings including dinucleotide composition (PseDNC), pseudo k-tuple nucleotide composition (PseKNC), parallel correlation pseudo dinucleotide composition (PC-PseDNC), parallel correlation pseudo trinucleotide composition (PC-PseTNC), series correlation pseudo dinucleotide composition (SC-PseDNC), and series correlation pseudo trinucleotide composition (SC-PseTNC) can be calculated by the package. The used 148 physicochemical properties for DNA dinucleotide and 22 physicochemical properties for RNA dinucleotide are listed in **Tables S3** and **S4**, respectively. The used 12 physicochemical properties for DNA trinucleotide are listed in **Table S5**.

PseDNC (Pseudo dinucleotide composition)

The Pseudo Dinucleotide Composition (PseDNC) encoding (44) incorporate contiguous local sequence-order information and the global sequence-order information into the feature vector of the nucleotide sequence. The PseDNC encoding is defined:

$$D = [d_1, d_2, \dots, d_{16}, d_{16+1}, \dots, d_{16+1}, \dots, d_{16+\lambda}]^T,$$

where

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda}\theta_j}, & (1 \le k \le 16) \\ \dfrac{w\theta_{k-16}}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda}\theta_j}, & (17 \le k \le 16+\lambda) \end{cases},$$

where f_k (k=1, 2, …, 16) is the normalized occurrence frequency of dinucleotide in the nucleotide sequence, λ represent the highest counted rank (or tie) of the correlation along the nucleotide sequence, w is the weight factor ranged from 0 to 1, and θ_j (j =1,2, …, λ) is the j-tier correlation factor and is defined:

$$\begin{cases} \theta_1 = \frac{1}{L-2}\sum_{i=1}^{L-2}\Theta(R_iR_{i+1}, R_{i+1}R_{i+2}) \\ \theta_2 = \frac{1}{L-3}\sum_{i=1}^{L-3}\Theta(R_iR_{i+1}, R_{i+2}R_{i+3}) \\ \theta_3 = \frac{1}{L-4}\sum_{i=1}^{L-4}\Theta(R_iR_{i+1}, R_{i+3}R_{i+4}) \quad (\lambda < L) \\ \cdots \\ \theta_\lambda = \frac{1}{L-1-\lambda}\sum_{i=1}^{L-1-\lambda}\Theta(R_iR_{i+1}, R_{i+\lambda}R_{i+\lambda+1}) \end{cases}$$

where the correlation function is defined:

$$\Theta\left(R_iR_{i+1}, R_{j+1}R_{j+1}\right) = \frac{1}{\mu}\sum_{u=1}^{\mu}[P_u(R_iR_{i+1}) - P_u\left(R_jR_{j+1}\right)]^2$$

where μ is the number of physicochemical indices. $P_u(R_iR_{i+1})$ is the numerical value of the u-th (u=1, 2, …, μ) physicochemical index of the dinucleotide R_iR_{i+1} at position i and $P_u\left(R_jR_{j+1}\right)$ represents the corresponding value of the dinucleotide R_jR_{j+1} at position j. The PseDNC descriptor has been successfully applied to recombination spot identification (44).

PseKNC (Pseudo *k*-tupler composition)

The Pseudo *k*-tupler Composition (PseKNC) encoding (45) incorporate the *k*-tuple nucleotide composition, which can be defined as:

$$D = [d_1, d_2, \ldots, d_{4^k}, d_{4^k+1}, \ldots, d_{4^k+\lambda}]^T,$$

where

$$\begin{cases} \frac{f_u}{\sum_{i=1}^{4^k} f_i + w\sum_{j=1}^{\lambda}\theta_j}, (1 \le u \le 4) \\ \frac{w\theta_{u-4^k}}{\sum_{i=1}^{4^k} f_i + w\sum_{j=1}^{\lambda}\theta_j}, (4^k \le u \le 4^k + \lambda) \end{cases},$$

where λ is the number of the total counted ranks (or tiers) of the correlations along a nucleotide sequence; f_u (u=1,2,…,4^k) is the frequency of oligonucleotide that is normalized to $\sum_{i=1}^{4^k} f_i = 1$, w is the factor, and θ_j is defined:

$$\theta_j = \frac{1}{L-j-1}\sum_{i=1}^{L-j-1}\Theta\left(R_iR_{i+1}, R_{i+j}R_{i+j+1}\right), \ (j = 1,2,\ldots,\lambda; \lambda < L).$$

The correlation function $\Theta\left(R_iR_{i+1}, R_{i+j}R_{i+j+1}\right)$ is defined as:

$$\Theta\left(R_iR_{i+1}, R_{i+j}R_{i+j+1}\right) = \frac{1}{\mu}\sum_{v=1}^{\mu}\left[P_v(R_iR_{i+1}) - P_v\left(R_{i+j}R_{i+j+1}\right)\right]^2,$$

where μ is the number of physicochemical indices. $P_v(R_iR_{i+1})$ is the numerical value of the v-th (v=1, 2, …, μ) physicochemical index of the dinucleotide R_iR_{i+1} at position i and $P_v\left(R_{j+j}R_{i+j+1}\right)$ represents the corresponding value of the dinucleotide $R_{i+j}R_{i+j+1}$ at position i+j. The PseKNC descriptor has been successfully applied to nucleosome positioning prediction (45).

PCPseTNC (Parallel correlation pseudo trinucleotide composition)

The Parallel Correlation Pseudo Trinucleotide Composition (PCPseTNC) encoding (46,47) is defined as:

$$D = [d_1, d_2, \dots, d_{64}, d_{64+1}, \dots, d_{64+\lambda}]^T,$$

where

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{64} f_i + w\sum_{j=1}^{\lambda}\theta_j}, (1 \le k \le 64) \\ \dfrac{w\theta_{k-64}}{\sum_{i=1}^{64} f_i + w\sum_{j=1}^{\lambda}\theta_j}, (65 \le k \le 64+\lambda) \end{cases},$$

where f_k (k=1, 2, …, 64) is the normalized occurrence frequency of trinucleotide in the DNA sequence, λ represent the highest counted rank (or tie) of the correlation along the DNA sequence, w is the weight factor ranged from 0 to 1, and θ_j (j =1,2, …, λ) is the j-tier correlation factor and is defined:

$$\begin{cases} \theta_1 = \frac{1}{L-3}\sum_{i=1}^{L-3}\Theta(R_iR_{i+1}R_{i+2}, R_{i+1}R_{i+2}R_{i+3}) \\ \theta_2 = \frac{1}{L-4}\sum_{i=1}^{L-4}\Theta(R_iR_{i+1}R_{i+2}, R_{i+2}R_{i+3}R_{i+4}) \\ \theta_3 = \frac{1}{L-5}\sum_{i=1}^{L-5}\Theta(R_iR_{i+1}R_{i+2}, R_{i+3}R_{i+4}R_{i+5}) \qquad (\lambda < L), \\ \dots \\ \theta_\lambda = \frac{1}{L-2-\lambda}\sum_{i=1}^{L-2-\lambda}\Theta(R_iR_{i+1}R_{i+2}, R_{i+\lambda}R_{i+\lambda+1}R_{i+\lambda+2}) \end{cases}$$

where the correlation function is defined:

$$\Theta\left(R_iR_{i+1}R_{i+2}, R_{j+1}R_{j+1}R_{j+2}\right) = \frac{1}{\mu}\sum_{u=1}^{\mu}\left[P_u(R_iR_{i+1}R_{i+2}) - P_u\left(R_jR_{j+1}R_{j+2}\right)\right]^2,$$

where μ is the number of physicochemical indices. $P_u(R_iR_{i+1}R_{i+2})$ is the numerical value of the u-th (u=1, 2, …, μ) physicochemical index of the dinucleotide $R_iR_{i+1}R_{i+2}$ at position i and $P_u\left(R_jR_{j+1}R_{j+2}\right)$ represents the corresponding value of the dinucleotide $R_jR_{j+1}R_{j+2}$ at position j. The PCPseTNC descriptor has been successfully applied to recombination spot identification (47).

SCPseDNC (Series correlation pseudo dinucleotide composition)

The Series Correlation Pseudo Dinucleotide Composition (SCPseDNC) encoding (46) is defined as:

$$D = [d_1, d_2, \dots, d_{16}, d_{16+1}, \dots, d_{16+\lambda}, d_{16+\lambda+1}, \dots, d_{16+\lambda\Lambda}]^T,$$

where

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda}\theta_j}, & (1 \le k \le 16) \\ \dfrac{w\theta_{k-16}}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda\Lambda}\theta_j}, & (17 \le k \le 16+\lambda\Lambda) \end{cases},$$

where f_k (k=1, 2, …, 16) is the normalized occurrence frequency of dinucleotide in the nucleotide sequence, λ represent the highest counted rank (or tie) of the correlation along the nucleotide sequence, w is the weight factor ranged from 0 to 1, Λ is the number of physicochemical indices and θ_j (j =1,2, …, λ) is the j-tier correlation factor and is defined:

$$\begin{cases} \theta_1 = \frac{1}{L-3}\sum_{i=1}^{L-3} J_{i,i+1}^1 \\ \theta_2 = \frac{1}{L-3}\sum_{i=1}^{L-3} J_{i,i+1}^2 \\ \cdots \\ \theta_\Lambda = \frac{1}{L-3}\sum_{i=1}^{L-3} J_{i,i+1}^\Lambda \quad (\lambda < L-2), \\ \cdots \\ \theta_{\lambda\Lambda} = \frac{1}{L-\lambda-2}\sum_{i=1}^{L-\lambda-2} J_{i,i+\lambda}^{\Lambda-1} \\ \theta_{\lambda\Lambda} = \frac{1}{L-\lambda-2}\sum_{i=1}^{L-\lambda-2} J_{i,i+\lambda}^{\Lambda} \end{cases}$$

where the correlation function is defined:

$$\begin{cases} J_{i,i+m}^{\zeta} = P_u(R_iR_{i+1})P_u(R_{i+m}R_{i+m+1}) \\ \zeta = 1,2,\dots,\Lambda; m = 1,2,\dots,\lambda; i = 1,2,\dots,L-\lambda-2 \end{cases},$$

where μ is the number of physicochemical indices. $P_u(R_iR_{i+1})$ is the numerical value of the u-th (u=1, 2, …, μ) physicochemical index of the dinucleotide R_iR_{i+1} at position i and $P_u\left(R_jR_{j+1}\right)$

represents the corresponding value of the dinucleotide R_jR_{j+1} at position *j*. The SCPseDNC descriptor has been successfully applied to recombination spot identification (47).

SCPseTNC (Series correlation pseudo trinucleotide composition)

The Series Correlation Pseudo Trinucleotide Composition (SCPseTNC) encoding (46) is defined as:

$$D = [d_1, d_2, \dots, d_{64}, d_{64+1}, \dots, d_{64+\lambda}, d_{64+\lambda+1}, d_{64+\lambda+1}, \dots, d_{64+\lambda\Lambda}]^T,$$

where

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{64} f_i + w\sum_{j=1}^{\lambda}\theta_j}, (1 \le k \le 64) \\ \dfrac{w\theta_{k-64}}{\sum_{i=1}^{64} f_i + w\sum_{j=1}^{\lambda\Lambda}\theta_j}, (65 \le k \le 64+\lambda\Lambda) \end{cases},$$

where f_k (*k*=1, 2, …, 64) is the normalized occurrence frequency of trinucleotide in the DNA sequence, λ represent the highest counted rank (or tie) of the correlation along the DNA sequence, *w* is the weight factor ranged from 0 to 1, Λ is the number of physicochemical indices and θ_j (*j* =1,2, …, λ) is the *j*-tier correlation factor and is defined:

$$\begin{cases} \theta_1 = \frac{1}{L-4}\sum_{i=1}^{L-4} J_{i,i+1}^1 \\ \theta_2 = \frac{1}{L-4}\sum_{i=1}^{L-4} J_{i,i+1}^2 \\ \cdots \\ \theta_\Lambda = \frac{1}{L-4}\sum_{i=1}^{L-4} J_{i,i+1}^\Lambda \quad (\lambda < L-3), \\ \cdots \\ \theta_{\lambda\Lambda} = \frac{1}{L-\lambda-3}\sum_{i=1}^{L-\lambda-3} J_{i,i+\lambda}^{\Lambda-1} \\ \theta_{\lambda\Lambda} = \frac{1}{L-\lambda-3}\sum_{i=1}^{L-\lambda-3} J_{i,i+\lambda}^{\Lambda} \end{cases}$$

where the correlation function is defined:

$$\begin{cases} J_{i,i+m}^{\zeta} = P_u(R_iR_{i+1})P_u(R_{i+m}R_{i+m+1}R_{i+m+2}) \\ \zeta = 1,2,\dots,\Lambda; m = 1,2,\dots,\lambda; i = 1,2,\dots,L-\lambda-3 \end{cases},$$

where μ is the number of physicochemical indices. $P_u(R_iR_{i+1}R_{i+2})$ is the numerical value of the *u*-th (*u*=1, 2, …, μ) physicochemical index of the dinucleotide $R_iR_{i+1}R_{i+2}$ at position *i* and $P_u(R_jR_{j+1}R_{j+2})$ represents the corresponding value of the dinucleotide $R_jR_{j+1}R_{j+2}$ at position *j*.

13. Descriptions of feature descriptors for protein or peptide sequences

AAC (Amino acid composition)

The Amino Acid Composition (AAC) encoding calculates the frequency of each amino acid type in a protein or peptide sequence. The frequencies of all 20 natural amino acids (i.e. "ACDEFGHIKLMNPQRSTVWY") can be calculated as:

$$f(t) = \frac{N(t)}{N}, \quad t \in \{A, C, D, ..., Y\},$$

where *N(t)* is the number of amino acid type *t*, while *N* is the length of a protein or peptide sequence. The AAC descriptor has been successfully applied to nuclear receptor classification (48) and anti-cancer peptide prediction (31).

EAAC (Enhanced amino acid composition)

The Enhanced Amino Acid Composition (EAAC) feature calculates the AAC based on the sequence window of fixed length that continuously slides from the N- to C-terminus of each peptide and can be usually applied to encode the peptides with an equal length (49). The EAAC can be calculated as:

$$f(t,\ win) = \frac{N(t, win)}{N(win)}, \quad t \in \{A, C, D, ..., Y\}, \quad win \in \{window1, window2, ..., window17\},$$

where *N(t,win)* is the number of amino acid type *t* in the sliding window *win* and *N(win)* is the size of the sliding window *win*. The EAAC descriptor has been successfully applied to protein malonylation sites prediction (50), lysine crotonylation site prediction (51).

CKSAAP (Composition of *k*-spaced amino acid pairs)

The CKSAAP feature encoding calculates the frequency of amino acid pairs separated by any *k* residues (*k* = 0, 1, 2, … , 5) (52-55). Taking *k* = 0 as an example, there are 400 0-spaced residue pairs (i.e., AA, AC, AD,…, YY.). Then, a feature vector can be defined as:

$$(\frac{N_{AA}}{N_{total}}, \frac{N_{AC}}{N_{total}}, \frac{N_{AD}}{N_{total}}, ..., \frac{N_{YY}}{N_{total}})_{400}.$$

The value of each descriptor denotes the composition of the corresponding residue pair in the protein

or peptide sequence. For instance, if the residue pair AA appears *m* times in the protein, the composition of the residue pair AA is equal to *m* divided by the total number of *0*-spaced residue pairs (N_{total}) in the protein. For $k = 0, 1, 2, 3, 4$ and 5, the value of N_{total} is $P - 1$, $P - 2$, $P - 3$, $P - 4$, $P - 5$ and $P - 6$ for a protein of length P, respectively. The EAAC descriptor has been successfully applied to membrane protein type prediction (52), protein crystallization prediction (53), protein flexible/rigid region prediction (54) and protein structural class prediction (55).

DPC (Di-Peptide Composition)

The Dipeptide Composition (56) gives 400 descriptors. It is defined as:

$$D(r,s) = \frac{N_{rs}}{N-1}, \qquad r,s \in \{A,C,D,\ldots Y\} \text{ ,}$$

where N_{rs} is the number of dipeptides represented by amino acid types *r* and *s.*

Dipeptide Deviation from Expected Mean (DDE)

The Dipeptide Deviation from Expected Mean feature vector is constructed by computing three parameters, i.e. dipeptide composition (D_c), theoretical mean (T_m), and theoretical variance (T_v). The above three parameters and the DDE are computed as follows. $D_c(r,s)$, the dipeptide composition measure for the dipeptide '*rs'*, is given as

$$D_c(r,s) = \frac{N_{rs}}{N-1}, \qquad r,s \in \{A,C,D,\ldots Y\} \text{ ,}$$

where N_{rs} is the number of dipeptides represented by amino acid types *r* and *s* and *N* is the length of the protein or peptide. $T_m(r,s)$, the theoretical mean, is given by:

$$T_m(r,s) = \frac{C_r}{C_N} \times \frac{C_s}{C_N},$$

where C_r is the number of codons that code for the first amino acid and C_s is the number of codons that code for the second amino acid in the given dipeptide '*rs*'. C_N is the total number of possible codons, excluding the three stop codons (i.e., 61). T_v (*r,s*), the theoretical variance of the dipeptide '*rs*', is given by:

$$T_v(r,s) = \frac{T_m(r,s)(1-T_m(r,s))}{N-1}.$$

Finally, *DDE(r,s)* is calculated as:

$$DDE(r,s) = \frac{D_c(r,s) - T_m(r,s)}{\sqrt{T_v(r,s)}}.$$

The DDE descriptor has been successfully applied to B-cell epitope prediction (56).

TPC (Tri-peptide composition)

The Tripeptide Composition (TPC) gives 8000 descriptors, defined as:

$$f(r,s,t) = \frac{N_{rst}}{N-2}, \qquad r,s,t \in \{A,C,D,...,Y\},$$

where N_{rst} is the number of tripeptides represented by amino acid types *r*, *s* and *t.*

Binary

In the binary encoding (57,58), each amino acid is represented by a 20-dimensional binary vector, e.g.

A is encoded by (10000000000000000000), C is encoded by (01000000000000000000), …, Y is encoded by (00000000000000000001), respectively. This encoding scheme is often used to encode peptides with an equal length. The binary descriptor has been successfully applied to protein ubiquitination sites prediction (57,58), kinase-specific phosphorylation site prediction (59).

Binary_6bit

In this descriptor, the 6-letter exchange group $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ is adopted to represent a protein sequence (14,60), where $e_1 \in$ {H, R, K}, $e_2 \in$ {D, E, N, D}, $e_3 \in$ {C}, $e_4 \in$ {S, T, P, A, G}, $e_5 \in$ {M, I, L, V}, $e_6 \in$ {F, Y, W}. Exchange groups represent conservative replacements through evolution. These exchange groups are effectively equivalence classes of amino acids and are derived from PAM. For example, the protein sequence PVKTNVK can be represented as $e_4e_5e_1e_4e_2e_5e_1$. Then, each group is represented by a 6-dimensional binary vector, e.g. e_1 is encoded by (100000), e_2 is encoded by (010000), …, e_6 is encoded by (000001).

Binary_5bit_type 1

Like binary_6bit descriptor, the 5-letter amino acid group $\{e_1, e_2, e_3, e_4, e_5\}$ is adopted to represent

a protein sequence, and each group is represented by a 5-dimensional binary vector (14,61). $e_1 \in$ {G, A, V, L, M, I}, $e_2 \in$ {F, Y, W}, $e_3 \in$ {K, R, H}, $e_4 \in$ {D, E}, $e_5 \in$ {S, T, C, P, N, Q}. Then, each group is represented by a 5-dimensional binary vector, e.g. e_1 is encoded by (10000), e_2 is encoded by (01000), …, e_5 is encoded by (00001).

Binary_5bit_type 2

For this descriptor, it is based on all the possible ways that ones and zeros can be combined in a five bit unit. There are 32 possible ways to represent 20 amino acids. When the representations with no or all ones and those with 1 or 4 ones are removed there are exactly twenty representations. And A is encoded by (00011), C (00101), D (00110), E (00111), F(01001), G (01010), H (01011), I (01100), K (01101), L (01110), M (10001), N (10010), P (10011), Q (10100), R (10101), S (10110), T (11000), V (11001), W (11010), Y (11100).

Binary_3bit_type 1

For this descriptor, the 3-letter amino acid group {e_1, e_2, e_3} is adopted to represent a protein sequence, and each group is represented by a 3-dimensional binary vector (31). Then, each group is represented by a 5-dimensional binary vector, e.g. e_1 is encoded by (100), e_2 is encoded by (010), e_3 is encoded by (001). The division of the amino acids based on physicochemical properties (PRAM900101) in **Table S6**. The difference of the type 1 to type 7 subtypes is the different division of amino acids. The division of amino acids for type 2 to 7 is based on “Normalized van der Waals volume”, “Polarity”, “Polarizability”, “Charge”, “Secondary structure” and “Solvent accessibility” in **Table S6**.

Table S6. Amino acid physicochemical attributes and the division of the amino acids into three groups according to each attribute.

Attribute	Division		
Hydrophobicity_PRAM900101	Polar: RKEDQN	Neutral: GASTPHY	Hydrophobicity: CLVIMFW
Hydrophobicity_ARGP820101	Polar: QSTNGDE	Neutral: RAHCKMV	Hydrophobicity: LYPFIW
Hydrophobicity_ZIMJ680101	Polar: QNGSWTDERA	Neutral: HMCKV	Hydrophobicity: LPFYI
Hydrophobicity_PONP930101	Polar: KPDESNQT	Neutral: GRHA	Hydrophobicity: YMFWLCVI
Hydrophobicity_CASG920101	Polar: KDEQPSRNTG	Neutral: AHYMLV	Hydrophobicity: FIWC
Hydrophobicity_ENGD860101	Polar: RDKENQHYP	Neutral :SGTAW	Hydrophobicity: CVLIMF
Hydrophobicity_FASG890101	Polar: KERSQD	Neutral: NTPG	Hydrophobicity: AYHWVMFLIC
Normalized van der Waals volume	Volume range: 0-2.78 GASTPD	Volume range: 2.95-94.0 NVEQIL	Volume range: 4.03-8.08 MHKFRYW
Polarity	Polarity value: 4.9-6.2 LIFWCMVY	Polarity value: 8.0-9.2 PATGS	Polarity value: 10.4-13.0 HQRKNED
Polarizability	Polarizability value: 0-1.08 GASDT	Polarizability value: 0.128-120.186 GPNVEQIL	Polarizability value: 0.219-0.409 KMHFRYW
Charge	Positive: KR	Neutral: ANCQGHILMFPSTWYV	Negative: DE
Secondary structure	Helix: EALMQKRH	Strand: VIYCWFT	Coil: GNPSD
Solvent accessibility	Buried: ALFCGIVW	Exposed: PKQEND	Intermediate: MPSTHY

AESNN3 (Learn from alignments)

For this descriptor, each amino acid type is described using a three-dimensional vector. Values are taken from the three hidden units from the neural network trained on structure alignments (14,62). The values are listed in **Table S7**.

Table S7. AESNN3 values learning from alignments.

Amino acids	AESNN3 values		
A	-0.99	-0.61	0.00
R	0.28	-0.99	-0.22
N	0.77	-0.24	0.59
D	0.74	-0.72	-0.35
C	0.34	0.88	0.35
Q	0.12	-0.99	-0.99
E	0.59	-0.55	-0.99
G	-0.79	-0.99	0.10
H	0.08	-0.71	0.68
I	-0.77	0.67	-0.37
L	-0.92	0.31	-0.99
K	-0.63	0.25	0.50
M	-0.80	0.44	-0.71
F	0.87	0.65	-0.53
P	-0.99	-0.99	-0.99
S	0.99	0.40	0.37
T	0.42	0.21	0.97
W	-0.13	0.77	-0.90
Y	0.59	0.33	-0.99
V	-0.99	0.27	-0.52

GAAC (Grouped amino acid composition)

In the GAAC encoding, the 20 amino acid types are further categorized into five classes according to their physicochemical properties, e.g. hydrophobicity, charge and molecular size (63). The five classes include the aliphatic group (*g1*: GAVLMI), aromatic group (*g2*: FYW), positive charge group (*g3*: KRH), negative charged group (*g4*: DE) and uncharged group (*g5*: STCPNQ). GAAC descriptor is the frequency of each amino acid group, which is defined as (8,49):

$$f(g) = \frac{N(g)}{N}, \quad g \in \{g1, g2, g3, g4, g5\}$$

$$N(g_t) = \sum N(t), \quad t \in g$$

where *N(g)* is the number of amino acid in group *g*, *N(t)* is the number of amino acid type *t,* and *N* is the length of the protein/peptide sequence.

EGAAC (Enhanced grouped amino acid composition)

The Enhanced GAAC (EGAAC) is also for the first time proposed in this work. It calculates GAAC in windows of fixed length continuously sliding from the N- to C-terminal of each peptide (8,49) and is usually applied to peptides with an equal length.

$$f(g,\ win) = \frac{N(g, win)}{N(win)}, \quad g \in \{g1, g2, g3, g4, g5\}, \quad win \in \{window1, window2, ..., window17\},$$

where *N(g, win)* is the number of amino acids in group *g* within the sliding window *win* and *N(win)* is the size of the sliding window *win*. The EGAAC descriptor has been successfully applied to lysine crotonylation sites prediction (51).

CKSAAGP (Composition of k-spaced amino acid group pairs)

The Composition of *k*-Spaced Amino Acid Group Pairs (CKSAAGP) is a variation of the CKSAAP descriptor, which is our own proposal. It calculates the frequency of amino acid group pairs separated by any *k* residues. Taking $k = 0$ as an example, there are 25 0-spaced group pairs (i.e., *g1g1, g1g2, g1g3, ... g5g5*). Thus, a feature vector of CKSAAGP can be defined as (8,49):

$$(\frac{N_{g1g1}}{N_{total}}, \frac{N_{g1g2}}{N_{total}}, \frac{N_{g1g3}}{N_{total}}, ..., \frac{N_{g5g5}}{N_{total}})_{25}.$$

The value of each descriptor denotes the composition of the corresponding residue group pair in a protein or peptide sequence. For instance, if the residue group pair *g1g1* appears *m* times in the protein, the composition of the residue pair *g1g1* is equal to *m* divided by the total number of *0*-spaced residue pairs (N_{total}) in the protein. For $k = 0, 1, 2, 3, 4$ and 5, the values of N_{total} are $P - 1$, $P - 2$, $P - 3$, $P - 4$, $P - 5$ and $P - 6$ respectively, for a protein of length *P*.

GDPC (Grouped dipeptide composition)

The Grouped Di-Peptide Composition encoding is another variation of the DPC descriptor. It is composed of a total of 25 descriptors that are defined as (8,49):

$$f(r,s) = \frac{N_{rs}}{N-1}, \quad r, s \in \{g1, g2, g3, g4, g5\},$$

where N_{rs} is the number of tripeptides represented by amino acid type groups *r* and *s*, and *N* is the length of a protein or peptide sequence.

GTPC (Grouped tripeptide composition)

The Grouped Tri-Peptide Composition encoding is also a variation of TPC descriptor, which generates 125 descriptors, defined as (8,49):

$$f(r,s,t)=\frac{N_{rst}}{N-2}, \quad r,s,t\in\{g1,g2,g3,g4,g5\},$$

where N_{rst} is the number of tripeptides represented by amino acid type groups r, s and t. N is the length of a protein or peptide sequence.

AAIndex

Physicochemical properties of amino acids are the most intuitive features for representing biochemical reactions and have been extensively applied in bioinformatics research. The amino acid indices (AAIndex) database (40) collects many published indices representing physicochemical properties of amino acids. For each physicochemical property, there is a set of 20 numerical values for all amino acids. Currently, 544 physicochemical properties can be retrieved from the AAIndex database. After removing physicochemical properties with value 'NA' for any of the amino acids, 531 physicochemical properties were left. In contrast to the residue-based encoding methods of amino acid identity and evolutionary information, a vector of 531 mean values is used to represent a sample for various window sizes. The AAIndex descriptor (64) can be applied to encode peptides of equal length. The AAIndex descriptor has been applied to protein ubiquitination site prediction (64) and protein malonylation site prediction (50).

ZScale

For this descriptor, each amino acid is characterized by five physicochemical descriptor variables (cf. **Table S8**), which were developed by Sandberg *et al.* in 1998 (65). The ZSCALE descriptor can be applied to encode peptides with equal length. The descriptor has been successfully applied to sumoylation site prediction (66).

Table S8. Z-scales for the 20 amino acids.

Amino acid	Z1	Z2	Z3	Z4	Z5	Amino Acid	Z1	Z2	Z3	Z4	Z5
A	0.24	-2.32	0.60	-0.14	1.30	M	-2.85	-0.22	0.47	1.94	-0.98
C	0.84	-1.67	3.71	0.18	-2.65	N	3.05	1.60	1.04	-1.15	1.61
D	3.98	0.93	1.93	-2.46	0.75	P	-1.66	0.27	1.84	0.70	2.00
E	3.11	0.26	-0.11	-3.04	-0.25	Q	1.75	0.50	-1.44	-1.34	0.66
F	-4.22	1.94	1.06	0.54	-0.62	R	3.52	2.50	-3.50	1.99	-0.17
G	2.05	4.06	0.36	-0.82	-0.38	S	2.39	-1.07	1.15	-1.39	0.67
H	2.47	1.95	0.26	3.90	0.09	T	0.75	-2.18	-1.12	-1.46	-0.40
I	-3.89	-1.73	-1.71	-0.84	0.26	V	-2.59	-2.64	-1.54	-0.85	-0.02
K	2.29	0.89	-2.49	1.49	0.31	W	-4.36	3.94	0.59	3.44	-1.59
L	-4.28	-1.30	-1.49	-0.72	0.84	Y	-2.54	2.44	0.43	0.04	-1.47

BLOSUM62

In this descriptor, the BLOSUM62 matrix is employed to represent the protein primary sequence information as the basic feature set. A matrix comprising of $m \times n$ elements is used to represent each residue in a training dataset, where n denotes the peptide length and $m = 20$, which elements comprise 20 amino acids. Each row in the BLOSUM62 matrix is adopted to encode one of 20 amino acids. The BLOSUM62 descriptor can be applied to encode peptides of equal length. The BLOSUM62 descriptor has been successfully applied to ubiquitination site prediction (67).

Moran

The autocorrelation descriptors are defined based on the distribution of amino acid properties along the sequence (68-70). The amino acid properties used here are different types of amino acids index, which is retrieved from the AAindex Database (40) available at http://www.genome.jp/dbget/aaindex.html/. All the amino acid indices are centralized and standardized prior to the calculation:

$$P_r = \frac{P_r - \overline{P}}{\sigma},$$

where $\overline{P}$ is the average of the properties of the 20 amino acids and σ is the standard deviation of the properties of the 20 amino acids. $\overline{P}$ and σ can be calculated as follows:

$$\overline{P} = \frac{\sum_{r=1}^{20} P_r}{20}, \quad \sigma = \sqrt{\frac{1}{20}\sum_{r=1}^{20}(P_r - \overline{P})^2}.$$

The Moran autocorrelation descriptors (68,71) can thus be defined as:

$$I(d) = \frac{\frac{1}{N-d}\sum_{i=1}^{N-d}(P_i - \overline{P}')(P_{i+d} - \overline{P}')}{\frac{1}{N}\sum_{i=1}^{N}(P_i - \overline{P}')^2}, \quad d = 1, 2, 3..., nlag,$$

where d is the lag of the autocorrelation, $nlag$ is the maximum value of the lag, P_i and P_{i+d} are the properties of the amino acids at positions i and $i + d$, respectively. $\overline{P}'$ is the average of the considered property P over the entire sequence of length N and is calculated as:

$$\overline{P}' = \frac{\sum_{i=1}^{N} P_i}{N}.$$

The Moran descriptor has been successfully applied to membrane protein type prediction (68) and protein secondary structural content prediction (71).

Geary

The Geary autocorrelation descriptors for a protein or peptide sequence are defined as:

$$C(d) = \frac{\frac{1}{2(N-d)}\sum_{i=1}^{N-d}(P_i - P_{i+d})^2}{\frac{1}{N-1}\sum_{i=1}^{N}(P_i - \overline{P}')^2}, \quad d = 1, 2, ..., nlag,$$

where d, P, P_i and P_{i+d}, $nlag$ have the same definitions as described above. The Geary descriptor has been successfully applied to population structure inferring (70).

NMBroto (Normalized moreau-broto autocorrelation)

The Moreau-Broto autocorrelation descriptors are defined as follows:

$$AC(d) = \sum_{i=1}^{N-d} P_i \times P_{i+d}, \qquad d = 1, 2, ..., nlag .$$

The normalized Moreau-Broto autocorrelation descriptors are thus defined as:

$$ATS(d) = \frac{AC(d)}{N-d}, \qquad d = 1, 2, ..., nlag .$$

The NMBroto descriptor has been successfully applied to protein helix content prediction (69).

Composition/Transition/Distribution (CTD)

The Composition, Transition and Distribution (CTD) features represent the amino acid distribution patterns of a specific structural or physicochemical property in a protein or peptide sequence (72-76). 13 types of physicochemical properties have been previously used for computing these features (**Table S6**). These include hydrophobicity, normalized Van der Waals Volume, polarity, polarizability, charge, secondary structures and solvent accessibility. These descriptors are calculated according to the following procedures: (i) The sequence of amino acids is transformed into a sequence of certain structural or physicochemical properties of residues; (ii) Twenty amino acids are divided into three groups for each of the seven different physicochemical attributes based on the main clusters of the amino acid indices of Tomii and Kanehisa (77). The groups of amino acids are listed in **Table S6**. The Composition/Transition/Distribution descriptors has been successfully applied to protein folding class prediction (72,73), enzyme family classification (75), RNA-binding protein prediction (76), protein structural prediction (77) and anti-cancer peptide prediction (31).

CTDC

Taking the hydrophobicity attribute as an example, all amino acids are divided into three groups: polar, neutral and hydrophobic (**Table S6**). The Composition descriptor consists of three values: the global compositions (percentage) of polar, neutral and hydrophobic residues of the protein. An illustrated example of this encoding scheme is provided in the following **Figure S22**. The Composition descriptor can be calculated as follows:

$$C(r) = \frac{N(r)}{N}, \quad r \in \{polar, neutral, hydrophobic\},$$

where *N(r)* is the number of amino acid type *r* in the encoded sequence and *N* is the length of the sequence.

CTDT

The Transition descriptor T also consists of three values (72,73): A transition from the polar group to the neutral group is the percentage frequency with which a polar residue is followed by a neutral residue or a neutral residue by a polar residue. Transitions between the neutral group and the hydrophobic group and those between the hydrophobic group and the polar group are defined in a similar way. The transition descriptor can then be calculated as:

$$T(r,s) = \frac{N(r,s) + N(s,r)}{N-1}, \quad r,s \in \{(polar, neutral), (neutral, hydrophobic), (hydrophobic, polar)\},$$

where *N(r,s)* and *N(s,r)* are the numbers of dipeptides encoded as "*rs*" and "*sr*" respectively in the sequence, while *N* is the length of the sequence. An illustrated example of this encoding scheme is provided in the following **Figure S22.**

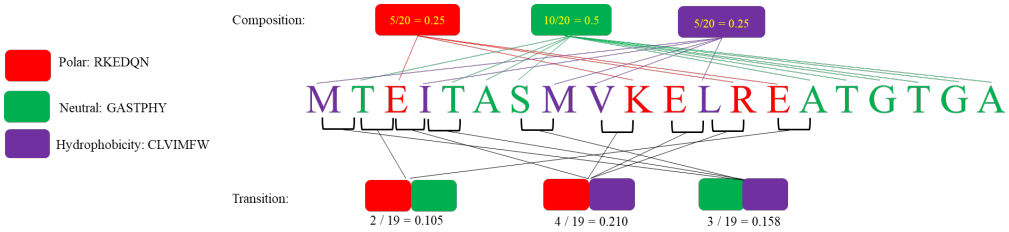


Figure S22. An example of the calculation of composition and transition descriptors. This example uses the hydrophobicity attribute.

CTDD

The Distribution descriptor consists of five values for each of the three groups (polar, neutral and hydrophobic) (72,73), namely the corresponding fraction of the entire sequence, where the first residue of a given group is located, and where 25, 50, 75 and 100% of occurrences are contained. For example, we start with the first residue up to and including the residue that marks 25/50/75/100% of occurrences for residues of any given group and then we simply divide the position of this residue by the length of the entire sequence.

CTriad (Conjoint triad)

The Conjoint Triad descriptor (CTriad) considers the properties of one amino acid and its vicinal amino acids by regarding any three continuous amino acids as a single unit (78). First, the protein sequence is represented by a binary space (*V, F*), where *V* denotes the vector space of the sequence features, and each feature (V_i) represents a sort of triad type; *F* is the number vector corresponding to *V*, where f_i, the value of the *i*-th dimension of *F*, is the number of type V_i appearing in the protein sequence. For the amino acids that have been catalogued into seven classes, the size of *V* should be equal to 7 x 7 x 7=343. Accordingly, i = 1, 2, 3, …, 343. An illustrated example of this encoding scheme is provided in the following **Figure S23**. In principle, the longer a protein sequence, the higher the probability to have larger values of f_i, confounding the comparison of proteins with different lengths. Thus, we define a new parameter, d_i, by normalizing f_i with the following equation:

$$d_i = \frac{f_i - \min\{f_1, f_2, ..., f_{343}\}}{\max\{f_1, f_2, ..., f_{343}\}}.$$

The CTriad descriptors has been successfully applied to protein-protein interaction prediction (78).

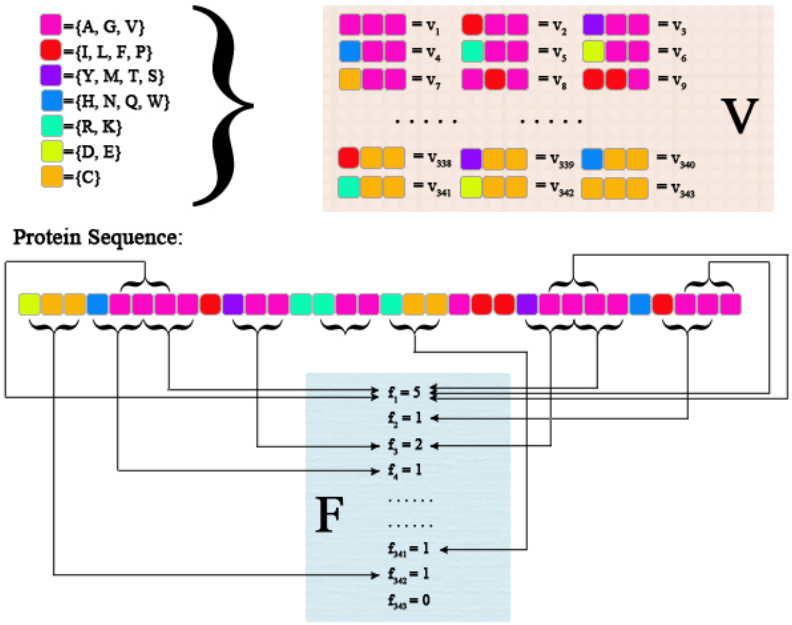


Figure S23. Schematic diagram for constructing the vector space (*V, F*) of a given protein sequence. *V* is the vector space of the sequence features; each feature (V_i) represents a triad composed of three consecutive amino acids; *F* is the number vector corresponding to *V*, and the value of the *i*-th entry of *F*, denoted f_i, is the number of occurrences that the triad associated with V_i appearing in the protein sequence. The figure was adapted from the Supplementary Figure in (78).

KSCTriad (Conjoint k-spaced Triad)

The *k*-Spaced Conjoint Triad (KSCTriad) descriptor is based on the Conjoint CTriad descriptor, which not only calculates the numbers of three continuous amino acid units, but also considers the continuous amino acid units that are separated by any *k* residues (The default maximum value of *k* is set to 5). For example, AxRxT is a 1-spaced triad. Thus, the dimensionality of the KSCTriad encoded feature vector is 343×(*k*+1). An illustrated example of this encoding scheme is provided in **Figure S24**.

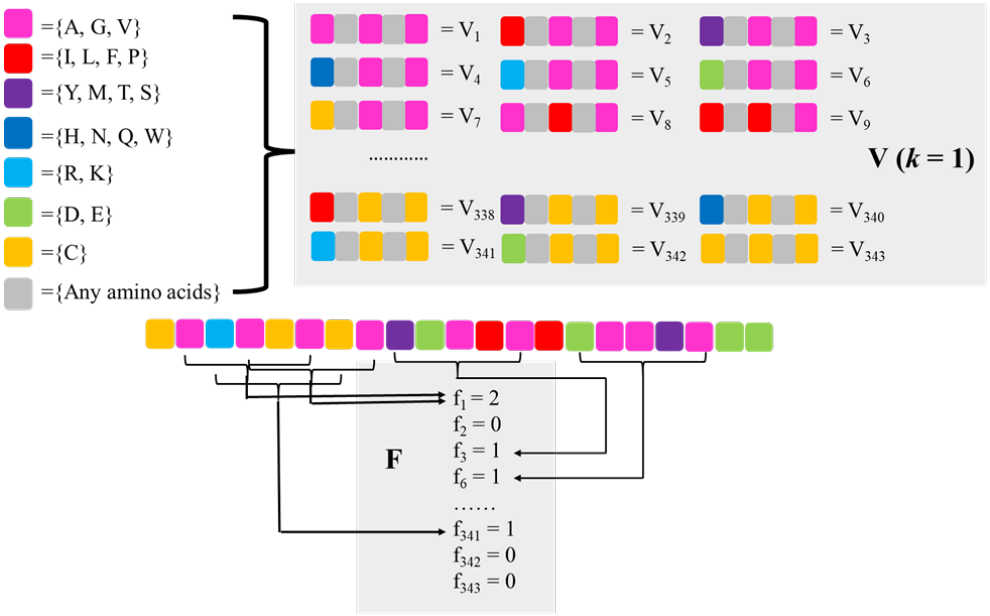


Figure S24. A schematic diagram for constructing the vector space (V, F) of protein sequence (*k*=1).

SOCNumber (Sequence-Order-Coupling Number)

The *d*-th rank sequence-order-coupling number is defined as:

$$\tau_d = \sum_{i=1}^{N-d} (d_{i,i+d})^2, \qquad d = 1,2,3,...,nlag,$$

where $d_{i,i+d}$ is the entry in a given distance matrix describing a distance between the two amino acids at position *i* and *i* + *d*, *nlag* denotes the maximum value of the lag (default value: 30) and *N* is the length of a protein or peptide sequence. As distance matrix both the Schneider-Wrede physicochemical distance matrix (79) used by Kuo-Chen Chou, and the chemical distance matrix by Grantham (80) are used. Accordingly, the descriptor dimension will be *nlag* x 2. The quasi-sequence-order descriptors described next also utilizes the two matrices. An illustrated example of this encoding scheme is provided in the following **Figure S25**.

Note: the length of the protein must be not less than the maximum value of *nlag*.

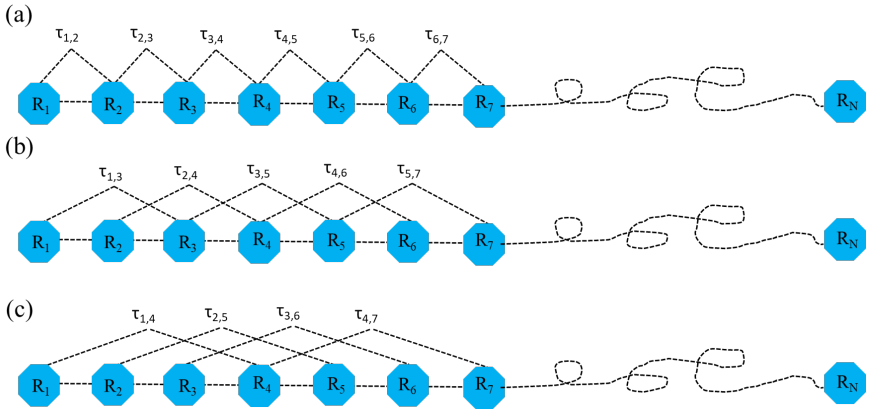


Figure S25. A schematic drawing to show (a) the 1st-rank, (b) the 2nd-rank, and (3) the 3rd-rank sequence-order-coupling mode along a protein sequence. (a) reflects the coupling mode between all the most adjacent residues, (b) shows the coupling between the adjacent plus one residue, and (c) shows the coupling between the adjacent plus two residues. This figure is adapted from (81).

QSOrder (Quasi-sequence-order)

For each amino acid type, a quasi-sequence-order descriptor can be defined as:

$$X_r = \frac{f_r}{\sum_{r=1}^{20} f_r + w\sum_{d=1}^{nlag} \tau_d}, \qquad r = 1, 2, ..., 20$$

where f_r is the normalized occurrence of amino acid type r and w is a weighting factor (w = 0.1), *nlag* and τ_d have the same definitions as described above. These are the first 20 quasi-sequence-order descriptors. The other 30 quasi-sequence-order descriptors are defined as:

$$X_d = \frac{w\tau_d - 20}{\sum_{r=1}^{20} f_r + w\sum_{d=1}^{nlag} \tau_d}, \qquad d = 21, 22, ..., 20 + nlag\ .$$

The SOCNumber and QSOrder descriptors have been successfully applied to protein subcellular location prediction (79,82).

PAAC (Pseudo-Amino Acid Composition)

This group of descriptors has been proposed in (83,84). Let $H_1^o(i)$, $H_2^o(i)$, $M^o(i)$ for i = 1, 2, 3, … 20 be the original hydrophobicity values, the original hydrophilicity values and the original side

chain masses of the 20 natural amino acids, respectively. They are converted to the following quantities by a standard conversion:

$$H_1(i) = \frac{H_1^o(i) - \frac{1}{20}\sum_{i=1}^{20} H_1^o(i)}{\sqrt{\frac{\sum_{i=1}^{20}[H_1^o(i) - \frac{1}{20}\sum_{i=1}^{20} H_1^o(i)]^2}{20}}},$$

where $H_2^o(i)$ and $M^o(i)$ are normalized as $H_2(i)$ and $M(i)$ in the same manner. An example of the correlation function is provided in the following **Figure S26**.

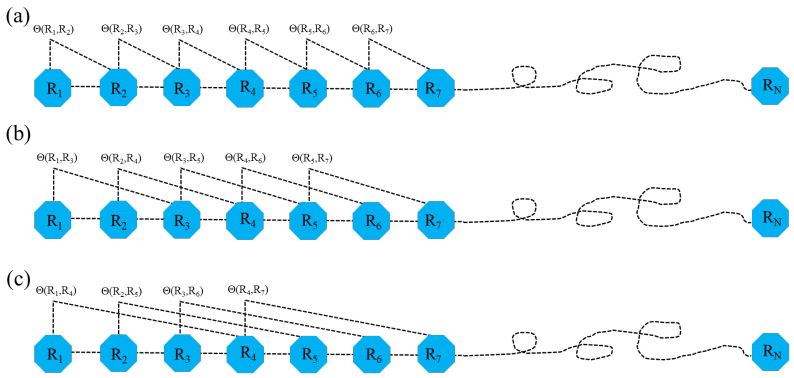


Figure S26. A schematic illustration showing (a) the first-tier, (b) the second-tier, and (3) the third-tier sequence order correlation mode along a protein sequence. (a) reflects the coupling mode between all the most adjacent residues, (b) shows the coupling between the adjacent plus one residue, and (c) shows the coupling between the adjacent plus two residues. This figure is adapted from (84) for illustration purposes.

Next, a correlation function can be defined as:

$$\Theta(R_i, R_j) = \frac{1}{3}\{[H_1(R_i) - H_1(R_j)]^2 + [H_2(R_i) - H_2(R_j)]^2 + [M(R_i) - M(R_j)]^2\}.$$

This correlation function is actually an averaged value for the three amino acid properties: hydrophobicity value, hydrophilicity value and side chain mass. Therefore, we can extend this definition of correlation function for one amino acid property or for a set of *n* amino acid properties. For one amino acid property, the correlation can be defined as:

$$\Theta(R_i, R_j) = [H_1(R_i) - H_1(R_j)]^2,$$

where $H(R_i)$ is the amino acid property of amino acid R_i after standardization.

For a set of n amino acid properties, it can be defined as:

$$\Theta(R_i, R_j) = \frac{1}{n}\sum_{n=1}^{n}[H_k(R_i) - H_k(R_j)]^2 ,$$

where $H_k(R_i)$ is the k-th property in the amino acid property set for amino acid R_i.

A set of descriptors called sequence order-correlated factors are defined as:

$$\theta_1 = \frac{1}{N-1}\sum_{i=1}^{N-1}\Theta(R_i, R_{i+1}) ,$$

$$\theta_2 = \frac{1}{N-2}\sum_{i=1}^{N-2}\Theta(R_i, R_{i+2}) ,$$

$$\theta_3 = \frac{1}{N-3}\sum_{i=1}^{N-3}\Theta(R_i, R_{i+3}) ,$$

$$\ldots,$$

$$\theta_\lambda = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda}\Theta(R_i, R_{i+\lambda}) ,$$

where λ ($\lambda < N$) is an integer parameter to be chosen. Let f_i be the normalized occurrence frequency of amino acid i in the protein sequence. Then, a set of $20 + \lambda$ descriptors called the pseudo-amino acid composition for a protein sequence can be defines as:

$$X_c = \frac{f_c}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{\lambda}\theta_j}, \qquad (1 < c < 20) ,$$

$$X_c = \frac{w\theta_{c-20}}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{\lambda}\theta_j}, \qquad (21 < c < 20 + \lambda) ,$$

where w is the weighting factor for the sequence-order effect and is set to $w = 0.05$ in *iLearnPlus* as suggested by Chou *et al*. (84).

APAAC (Amphiphilic Pseudo-Amino Acid Composition)

Amphiphilic Pseudo-Amino Acid Composition (APAAC) was proposed in (83,84). The definition of this set of features is similar to the PAAC descriptors. Using $H_1(i)$ and $H_2(j)$ as previously defined,

the hydrophobicity and hydrophilicity correlation functions are defined as:

$$H_{i,j}^1 = H_1(i)H_1(j)$$
$$H_{i,j}^2 = H_2(i)H_2(j)$$

respectively. An illustrated example of the correlation functions is provided in the following **Figure S27**. Thus, sequence order factors can be defined as:

$$\tau_1 = \frac{1}{N-1}\sum_{i=1}^{N-1} H_{i,i+1}^1$$
$$\tau_2 = \frac{1}{N-1}\sum_{i=1}^{N-1} H_{i,i+1}^2$$
$$\tau_3 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^1$$
$$\tau_4 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^2$$
...
$$\tau_{2\lambda-1} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^1$$
$$\tau_{2\lambda} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^2$$

Then, Amphiphilic Pseudo-Amino Acid Composition (APAAC) is defined as:

$$P_c = \frac{f_c}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{2\lambda} \tau_j}, \qquad (1 < c < 20)$$

$$P_c = \frac{\omega\tau_u}{\sum_{r=1}^{20} f_r + w\sum_{j=1}^{2\lambda} \tau_j}, \qquad (21 < u < 20 + 2\lambda)$$

where w is the weighting factor. In *iLearnPlus* this factor is set to $w = 0.5$ as described in Chou's work (84). The PAAC and APAAC have been successfully applied to protein cellular attributes prediction (84) and enzyme subfamily classes prediction (83).

OPF_10bit

For this descriptor, the amino acids are classified into 10 groups based their physicochemical properties (**Table S9**). Note that different groups might overlap, since a specific amino acid type may have two or more physicochemical properties. To reflect the correlation of different properties, a 10-bit vector were calculated to represent each amino acid of the N-terminus of peptide. Similarly,

the position of the bit of this 10-bit vector is set to 1, if the amino acid belongs to a corresponding group and 0 otherwise. The OPF_10bit descriptor has been successfully applied to anti-cancer peptides prediction (31).

OPF_7bit type 1

Similar to OPF_10bit descriptor, for this descriptor, the amino acids are classified into 7 groups. There are three subtypes of OPF_7bit descriptor (i.e. type 1, type 2 and type 3), due to different division of the amino acids. The difference of the type 1 to type 3 subtypes is the different division of amino acids. The division of amino acids for type 1 to 3 is listed in **Table S10**.

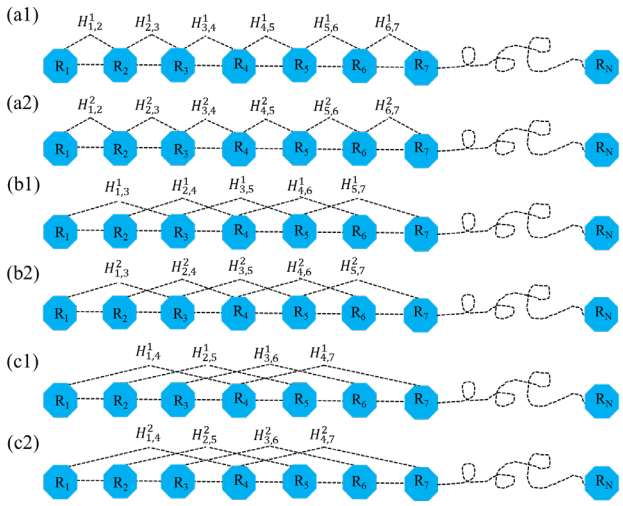


Figure S27. A schematic diagram to show (a1/a2) the first-rank, (b1/b2) the second-rank and (c1/c2) the third-rank sequence-order-coupling mode along a protein sequence through a hydrophobicity/hydrophilicity correlation function, where $H_{i,j}^{1}$ and $H_{i,j}^{2}$ are given by the aforementioned equation. Panels (a1/a2) reflects the coupling mode between the most adjacent residues, panels (b1/b2) shows the coupling between the adjacent plus one residue, and panels (c1/c2) shows the coupling between the adjacent plus two residues. This figure is adapted from (83) for illustration purposes.

Table S9. Details of the division of the standard amino acid alphabet based on ten physicochemical properties.

Rank	Physicochemical properties	Amino acid group
1	Aromatic	{F, Y, W, H}
2	Negative	{D, E}
3	Positive	{K, H, R}
4	Polar	{N, Q, S, D, E, C, T, K, R, H, Y, W}
5	Hydrophobic	{A, G, C, T, I, V, L, K, H, F, Y, W, M}
6	Aliphatic	{I, V, L}
7	Tiny	{A, S, G, C}
8	Charged	{K, H, R, D, E}
9	Small	{P, N, D, T, C, A, G, S, V}
10	Proline	{P}

Table S10. Details of the division of the standard amino acid alphabet based on seven physicochemical properties.

Rank	Physicochemical properties	Type 1	Type 2	Type 3
1	Hydrophobicity	{A, C, F, G, H, I, L, M, N, P, Q, S, T, V, W, Y}	{D, E}	{K, R}
2	Normalized Van der Waals volume	{C, F, I, L, M, V, W}	{A, G, H, P, S, T, Y}	{D, E, K, N, Q, R}
3	Polarity	{A, C, D, G, P, S, T}	{E, I, L, N, Q, V}	{F, H, K, M, R, W, Y}
4	Polarizibility	{C, F, I, L, M, V, W, Y}	{A, G, P, S, T}	{D, E, H, K, N, Q, R}
5	Charge	{A, D, G, S, T}	{C, E, I, L, N, P, Q, V}	{F, H, K, M, R, W, Y}
6	Secondary structures	{D, G, N, P, S}	{A, E, H, K, L, M, Q, R}	{C, F, I, T, V, W, Y}
7	Solvent accessibility	{A, C, F, G, I, L, V, W}	{H, M, P, S, T, Y}	{D, E, K, N, R, Q}

ASDC (Adaptive skip dinucleotide composition)

The adaptive skip dipeptide composition is a modified dipeptide composition, which sufficiently considers the correlation information present not only between adjacent residues but also between intervening residues (31). For given a sequence, the feature vector for ASDC is represented by:

$$ASDC = (f_{v1}, f_{v1}, \dots, f_{v400}),$$

where f_{vi} is calculated by

$$f_{vi} = \frac{\sum_{g=1}^{L-1} O_i^g}{\sum_{i=1}^{400} \sum_{g=1}^{L-1} O_i^g},$$

where f_{vi} denotes the occurrence frequency of all possible dipeptide with ≤ L-1 intervening nucleotides. The ASDC descriptor has been successfully applied to anti-cancer peptide prediction (31) and cell-penetrating peptide prediction (32).

KNN (K-Nearest Neighbor for peptides)

The *K*-Nearest Neighbor for peptides (KNN) descriptor depicts how much one query sample resembles other samples. Here, the similarity score between two peptides is defined as:

$$Score = \sum_{i=1}^{n} S(P_{1,i}, P_{2,i})$$

$$S(a,b) = \begin{cases} BLOSUM62(a,b), & if\ (BLOSUM62) > 0 \\ 0, & if\ (BLOSUM62) \leq 0 \end{cases}$$

where *P* is the peptide with *n* amino acids, *i* is the sequence position, and *BLOSUM62(a,b)* is the corresponding element value for amino acids *a* and *b* in the BLOSUM62 matrix. The KNN descriptor has been successfully applied to ubiquitination site prediction (85).

DistancePair (PseAAC of distance-pair and reduced alphabet)

The descriptor incorporates the amino acid distance pair coupling information and the amino acid reduced alphabet profile into the general pseudo amino acid composition vector. For the reduced alphabet profile, they are cp(13), cp(14), and cp(15) as defined below:

cp(13) = {MF; IL;V;A;C;WYQHP;G;T; S;N;RK;D; E}

cp(14) = {EIMV; L;F;WY;G; P;C;A; S; T;N;HRKQ; E;D}

sp(15) = {P;G; E;K;R;Q;D; S;N; T;H;C; I;V;W;YF;A; L;M}

where the single letters without a semicolon (;) to separate them mean belonging to a same cluster. The DistancePair descriptor has been successfully applied to DNA-binding protein identification (14).

PseKRAAC (pseudo K-tuple reduced amino acids composition)

Previous studies indicate that certain residues are similar in their physicochemical features, and can be clustered into groups because they play similar structural or functional roles in proteins (86). By implementing reduced amino acid alphabets, the protein complexity can be significantly simplified, which reduces information redundancy and decreases the risk of overfitting. The Pseudo *K*-tuple Reduced Amino Acids Composition (PseKRAAC) descriptor (87) includes two different feature types for protein sequence analysis: *g-gap* and λ-correlation PseKRAAC (84).

The *g-gap* PseKRAAC is used to represent a protein sequence with a vector containing $RAAC^K$ components, where *g* represents the gap between each *K*-tuple peptides (39,43,88,89). A *g-gap* of *n* reflects the sequence-order information for all *K*-tuple peptides with the starting residues separated by *n* residues. An illustrated example of this encoding scheme ($K = 2$) is provided in the following **Figure S28A**.

The *λ*-correlation PseKRAAC is used to represent a protein sequence with a vector containing $RAAC^K$ components, where *λ* is an integer that represents the correlation tier and is less than *N-K*, where *N* is the sequence length. The *n*-th-tier correlation factor ($\lambda = n$) reflects the sequence-order correlation between the *n*-th nearest residues. An illustrated example of this encoding scheme ($K = 2$) is provided in the following **Figure S28B**.

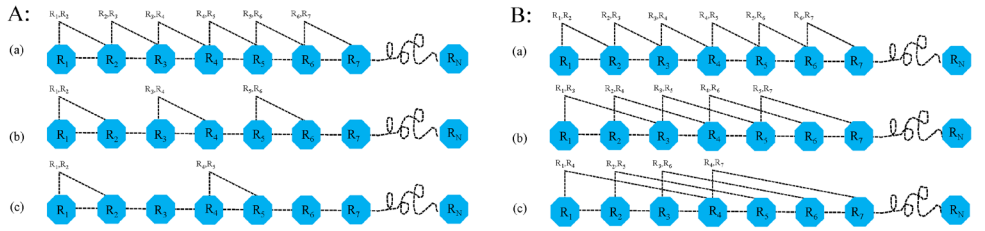


Figure S28. A schematic diagram showing: (A) *g-gap* definition of dipeptide, and (B) λ-correlation definition of dipeptide A: (a) *g-gap* of 0 reflects the sequence-order information between all adjacent dipeptides, i.e. separated by zero residues, (b) *g-gap* of 1 reflects the sequence-order information for all dipeptides with the starting residues separated by one residue, and (c) *g-gap* of 2 reflects the sequence-order information for all dipeptides with the starting residues separated by two residues; B: (a) the first-tier correlation factor reflects the sequence-order correlation between the nearest residues along a protein chain, (b) the second-tier correlation factor reflects the sequence-order correlation between the second nearest residues, (c) the third-tier correlation factor reflects the sequence-order correlation between the 3rd nearest residues, and so forth. The figure is adapted from (87).

The 16 types of reduced amino acid alphabets with different clustering approaches can be used to generate different versions of pseudo reduced amino acid compositions (PseRAACs) (**Table S11**).

Table S11. A list of 16 types of reduced amino acid alphabets for proteins (87).

Type	Description	Cluster	Reference
1	RedPSSM	2-19	(90)
2	BLOSUM 62 matrix	2-6, 8, 15	(91)
3	PAM matrix (3A) and WAG matrix (3B)	2-19	(92)
4	Protein Blocks	5,8,9,11,13	(93)
5	BLOSUM50 matrix	3,4,8,10,15	(93)
6	Multiple cluster	4,5A,5B,5C	(94)
7	Metric multi-dimensional scaling	2-19	(95)
8	Grantham Distance Matrix	2-19	(96)
9	Grantham Distance Matrix	2-19	(96)
10	BLOSUM matrix for SWISS-PROT	2-19	(97)
11	BLOSUM matrix for SWISS-PROT	2-19	(97)
12	BLOSUM matrix for DAPS	2-18	(98)
13	Coarse-graining substitution matrices	4,12,17	(99)
14	Alphabet Simplifer	2-19	(100)
15	MJ matrix	2-16	(101)
16	BLOSUM50 matrix	2-16	(101)

14. Guidance of parameters setting for conventional machine learning algorithms

For the conventional machine-learning algorithms, *iLearnPlus* supports automatic parameter optimization for the main parameters of these algorithms. Users who are not familiar with the parameters can use parameter(s) auto optimization function to find optimized parameter(s). For example, if you do not know how to set the “Penalty” and “Gamma” values of “rbf” kernel in SVM algorithm, you can select the “Auto optimization” option to optimize these two parameters automatically. We suggest using the default values when users are not familiar with a given parameter.

RF (Random forest)

RF (102) is a well-established and widely employed predictive/supervised algorithm which ensembles/combines of a given number of decision trees.

Parameters	*1*	*Tree number*:	The number of trees in the forest, default=100.
	2	*Number of threads*:	The number of threads to run in parallel, default=1.
	3	*Auto optimization*:	If "*Auto optimization*" is checked, the program will optimize parameter of "*Tree number*" automatically.
	4	*Trees range from*:	The searching space for "*Tree number*" parameters. If "*Auto optimization*" is checked, the program will test the performance with tree number value from "*Trees range from*" to "*Trees range to*" with steps as "*Tree step*".
	5	*Trees range to*:	
	6	*Tree step*:	

LightGBM

LightGBM (103) is a gradient boosting predictive/supervised method that uses tree-based learning algorithms.

Parameters	*1*	*Boosting type*:	Boosting type, default='gbdt'.
	2	*Number of leaves*:	Maximum tree leaves for base learners. This is the main parameter to control the complexity of the tree model. Should let it be small than $2^{Max\ depth}$, default=31.
	3	*Max depth*:	Maximum tree depth for base learners, <=0 means no limit.
	4	*Learning rate*:	Boosting learning rate, default=0.1.
	5	*Number of threads*:	The number of threads to run in parallel, default=1.
	6	*Auto optimization*:	If "*Auto optimization*" is checked, the program will optimize the parameters "*Number of leaves*", "*Max depth*" and "*Learning rate*" automatically.
	7	*Leaves range:*	Searching space for "*Number of leaves*", with the format of "from:to:step", default=20:100:10, when "*Auto optimization*:" is checked.
	8	*Depth range:*	Searching space for "*Max depth*", default=15:55:10, only when "*Auto optimization*:" is checked.
	9	*Learning rate range:*	Searching space for "*Learning rate*", default=0.01:0.15:0.02, only when "*Auto optimization*:" is checked.

SVM (Support Vector Machine)

SVM (104) is a predictive/supervised algorithm that accurately classifies samples by generating the optimal hyperplanes using quadratic programming on the training data. A variety of kernels can be

used to further improve predictive performance of SVM including Gaussian radial basis function (RBF), linear kernel, polynomial kernel, sigmoid kernel, etc.

Parameters	*1*	*Kernel function*	Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf' or 'sigmoid'.
	2	*Penalty*	Regularization parameter, default=1. The strength of the regularization is inversely proportional to *Penality*. Must be strictly positive.
	3	*Gamma*	Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. if 'auto', uses 1 / n_features.
	4	*Auto optimization*	If "*Auto optimization*" is checked, the program will optimize the parameters "*Penalty*" and "*Gamma*" automatically.
	5	*Penalty from*	Searching space for parameter "*Penalty*", default from 1.0 to 15.0
	6	*Penalty to*	
	7	*Gamma from*	Searching space for parameter "*Gamma*", default from 2^{-10} to 2^{5}.
	8	*Gamma to*	

MLP (Multi-layer Perceptron)

MLP (105) is a predictive/supervised algorithm that implements a network of neurons, typically involving three or more interconnected layers of neutorn, that are wired in a linear/feed-forward fashion.

Parameters	*1*	*Hidden layer size*	Integers separated by ":", The *i*-th element represents the number of neurons in the *i*-th hidden layer. For example, 32:64:2 means a neural network with three layers, with 32 neurons in first layer, 64 neurons in secondary layer and 2 neurons in the third layer.
	2	*Epochs*	Maximum number of iterations. The solver iterates until convergence or this number of iterations, default=200.
	3	*Activation*	Activation function for the hidden layer, default=relu.
	4	*Optimizer*	The solver for weight optimization, default=adam.

XGBoost

XGBoost (4) is a tree-boosting predictive/supervised algorithm which is an advanced implementation of the gradient boosting algorithm. Given its favorable predictive performance in a wide range of practical applications and recently-held competitions, it has been widely applied to solve many classification problems in recent years.

Parameters	*1*	*Booster*	Which booster to use. Can be gbtree, gblinear; gbtree use tree based models while gblinear uses linear functions.
	2	*Threads number*	The number of threads to run in parallel, default=1.
	3	*Max depth*	Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit, default=6.
	4	*Learning rate*	Step size shrinkage used in update to prevents overfitting, default=0.3.
	5	*colsample_bytree*	This the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed, default=0.8.
	6	*Auto optimization*	If "*Auto optimization*" is checked, the program will optimize the parameters "*Max depth*" and "*Learning rate*" automatically.
	7	*Depth range*	Searching space for parameter "*Max depth*", with the format of "from:to:step", default=3:10:1, only when "*Auto optimization*:" is checked.
	8	*Learning rate range*	Searching space for parameter "*Learning rate*", with the format of "from:to:step", default=0.01:0.3:0.05, only when "*Auto optimization*:" is checked.

KNN (K-Nearest Neighbor)

KNN (106) algorithm is a commonly employed supervised algorithm that relies on calculating similarities/distances between objects/sequences.

Parameters	*1*	*Top K values*	Number of neighbors to use by default for k-neighbors queries, default=3.

Bagging

Bagging (107) is a popular approach to combine/ensemble results produced by multiple supervised/predictive algorithms.

Parameters	*1*	*n_estimators*	The number of base estimators in the ensemble, default =10.
	2	*Number of threads*	The number of threads to run in parallel, default=1.

15. Guidance of parameters setting for deep learning algorithms

For Net_1_CNN, Net_2_RNN, Net_3_BRNN, Net_4_ABCNN, and Net_5_ResNet, the "Input channels" and "Input length" are regarded as two key parameters that determine the predictive performance of these models. These methods can use residue-level feature encoding as the input. The residue-level features refer to individual amino acids or nucleotides in the sequence being encoded into fixed-size feature vectors. For example, in binary encoding for protein sequences, each amino acid is encoded as a 20-dimension binary vector (one of the 20 bits is set 1 for the corresponding amino acid type while other bits are set to 0). Some other protein descriptors such as AAIndex, ZScale, BLOSUM62, OPF, and nucleotide descriptors like ANF, binary and EIIP are also residue-level descriptors. The "Input channels" is the vector dimension for each amino acids/nucleotide in the input sequence (e.g. Input channels=20 for the abovementioned binary encoding) while the "Input length" denotes the sequence length. Thus, the product of the "Input channels" and "Input length" determines the total numbers of input features for the given sequence. Importantly, these methods can also take the sequences-level feature encoding as the input. This would mean, for instance, that the input protein sequence could be encoded using amino acid composition (AAC) of the entire sequences. In this case, the users should set "Input channels" =1 and "Input length" = vector dimension for the sequence encoding (20 when using AAC). The descriptions of the remaining parameters are provided in the subsequent tables. We suggest using the default values when users are not familiar with a given parameter.

Net_1_CNN

Parameters	*1*	*Output channels*	The number of convolving kernels in CNN, default=64.
	2	*Padding*	Controls the amount of implicit zero-paddings on both sides for padding number of points in convolution operation, default=2.
	3	*Kernel size*	Size of the convolving kernel for CNN, default=5.
	4	*FC layer size*	The neurons number in flatten layer, default=64.
	5	*Dropout rate*	Dropout is a regularization technique which prevents over-fitting of the network. During training a certain number of neurons in the hidden layer is randomly dropped. Default=0.5.
	6	*Learning rate*	The learning rate is defined as the amount of minimization in the cost func+tion in each iteration, default=0.001.
	7	*Epochs*	A single training iteration of all batches in both forward and back

			propagation. This means 1 epoch is a single forward and backward pass of the entire input data, default=1000.
	8	*Early stopping*	If the performance did not improve within the value of "*Early stopping*" epochs, the model optimization process will be terminated early. Default=100.
	9	*Batch size*	While training a neural network, instead of sending the entire input in one go, we divide in input into several chunks of equal size randomly. Training the data on batches makes the model more generalized as compared to the model built when the entire data set is fed to the network in one go. Default=64.

Net_2_RNN and Net_3_BRNN

Parameters	*1*	*Hidden size*	The number of features in the hidden state *h*, default=32.
	2	*Number of recurrent layers*	Number of recurrent layers. E.g., setting num_layers=2 would mean stacking two LSTMs together to form a stacked LSTM, with the second LSTM taking in outputs of the first LSTM and computing the final results. Default: 1.

Please refer to Net_1_CNN for the description of the other parameters.

Net_4_ABCNN and Net_5_ResNet

Please refer to Net_1_CNN for the description of the parameters.

Net_6_AE

Parameters	*1*	*Input size*	The dimension of the feature encoding, the parameter will be determined automatically.

16. Guide to source code location

For the users' convenience, we provide a brief guide that users can utilize to identify location of a given part of the source code from *iLearnPlus*. The source code guide is summarized in **Table S12**.

Table S12. Guide to source code location for *iLearnPlus*.

Primary directory	Secondary directory	Content
data	*Txt files*	The example files are stored in this directory
docs	*iLearnPlus_manual.pdf*	User manual
images	*Image files*	Icon and image files
models	*Model files ended with ".pkl"*	The models will be saved in this directory by default
util	*CheckAccPseParameter.py*	Parameters validation for descriptors of DAC, DCC, DACC, PseDNC, PseKNC, PCPseDNC, PCPseTNC, SCPseDNC and SCPseTNC
	DataAnalysis.py	The class used for feature analysis including clustering, feature normalization, selection and dimensionality reduction
	FileProcessiong.py	The class used for feature descriptor extraction, which includes all the feature descriptor methods
	EvaluationMetrics.py	The class used for calculating evaluation metrics
	InputDialog.py	The GUI class for all the dialogs, which are used for parameters setting
	MachineLearning.py	The class used for model construction, include all the conventional machine learning algorithms
	MCL.py	The markov clustering algorithm
	ModelMetrics.py	The class used for storing the data of evaluation metrics, and plotting data for ROC and PRC curves.
	Modules.py	The GUI classes used for plotting ROC, PRC curves, scatter plot and kernel density plot by using the user-defined data.
	Nets.py	The architectures of the deep-learning algorithms are stored in this file
	PlotWidgets.py	The GUI classes used for plotting, such as ROC, PRC curves, scatter plot and kernel density
	TableWidget.py	The GUI for table display
iLearnPlus.py	--	The GUI for starting program
iLearnPlusBasic.py	--	The GUI for *iLearnPlus-Basic* module
iLearnPlusEstimator.py	--	The GUI for *iLearnPlus-Estimator* module
iLearnPlusAutoML.py	--	The GUI for *iLearnPlus-AutoML* module
iLearnPlusLoadModel.py	--	The GUI for *iLearnPlus-LoadModel* module

References

1. Chang, C.C. and Lin, C.J. (2011) LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, Article 27.
2. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009) The

WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, **11**, 10–18.
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. *et al.* (2011) Scikit-learn: Machine Learning in Python. *J Mach Learn Res*, **12**, 2825–2830.
4. Chen, T. and Guestrin, C. (2016) XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, San Francisco, California, USA, pp. 785–794.
5. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y. (2017) LightGBM: a highly efficient gradient boosting decision tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Long Beach, California, USA, pp. 3149–3157.
6. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. *et al.* (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alche-Buc, F., Fox, E. and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, Vol. 32. pp. 8024–8035.
7. Zhou, C., Wang, C., Liu, H., Zhou, Q., Liu, Q., Guo, Y., Peng, T., Song, J., Zhang, J., Chen, L. *et al.* (2018) Identification and analysis of adenine N(6)-methylation sites in the rice genome. *Nat Plants*, **4**, 554–563.
8. Chen, Z., Zhao, P., Li, F., Marquez-Lago, T.T., Leier, A., Revote, J., Zhu, Y., Powell, D.R., Akutsu, T., Webb, G.I. *et al.* (2020) iLearn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data. *Brief Bioinform*, **21**, 1047–1057.
9. Lee, D., Karchin, R. and Beer, M.A. (2011) Discriminative prediction of mammalian enhancers from DNA sequence. *Genome Res*, **21**, 2167–2180.
10. Noble, W.S., Kuehn, S., Thurman, R., Yu, M. and Stamatoyannopoulos, J. (2005) Predicting the in vivo signature of human gene regulatory sequences. *Bioinformatics*, **21 Suppl 1**, i338–343.
11. Sun, L., Luo, H., Bu, D., Zhao, G., Yu, K., Zhang, C., Liu, Y., Chen, R. and Zhao, Y. (2013) Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. *Nucleic Acids Res*, **41**, e166.
12. Wang, L., Park, H.J., Dasari, S., Wang, S., Kocher, J.P. and Li, W. (2013) CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model. *Nucleic Acids Res*, **41**, e74.
13. Gupta, S., Dennis, J., Thurman, R.E., Kingston, R., Stamatoyannopoulos, J.A. and Noble, W.S. (2008) Predicting human nucleosome occupancy from primary sequence. *PLoS Comput Biol*, **4**, e1000134.
14. Liu, B., Gao, X. and Zhang, H. (2019) BioSeq-Analysis2.0: an updated platform for analyzing DNA, RNA and protein sequences at sequence level and residue level based on machine learning approaches. *Nucleic Acids Res*, **47**, e127.
15. Leslie, C.S., Eskin, E., Cohen, A., Weston, J. and Noble, W.S. (2004) Mismatch string kernels for discriminative protein classification. *Bioinformatics*, **20**, 467–476.
16. El-Manzalawy, Y., Dobbs, D. and Honavar, V. (2008) Predicting flexible length linear B-cell

epitopes. *Comput Syst Bioinformatics Conf*, **7**, 121–132.
17. Luo, L., Li, D., Zhang, W., Tu, S., Zhu, X. and Tian, G. (2016) Accurate Prediction of Transposon-Derived piRNAs by Integrating Various Sequential and Physicochemical Features. *PLoS One*, **11**, e0153268.
18. Chen, W., Tran, H., Liang, Z., Lin, H. and Zhang, L. (2015) Identification and analysis of the N(6)-methyladenosine in the Saccharomyces cerevisiae transcriptome. *Sci Rep*, **5**, 13859.
19. Chen, Z., Zhao, P., Li, F., Wang, Y., Smith, A.I., Webb, G.I., Akutsu, T., Baggag, A., Bensmail, H. and Song, J. (2020) Comprehensive review and assessment of computational methods for predicting RNA post-transcriptional modification sites from RNA sequences. *Brief Bioinform*, **21**, 1676–1696.
20. He, X., Zhang, S., Zhang, Y., Jiang, T. and Zeng, J. (2017) Characterizing RNA Pseudouridylation by Convolutional Neural Networks. *bioRxiv*, 126979.
21. Alipanahi, B., Delong, A., Weirauch, M.T. and Frey, B.J. (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotechnol*, **33**, 831–838.
22. Zhou, J. and Troyanskaya, O.G. (2015) Predicting effects of noncoding variants with deep learning-based sequence model. *Nat Methods*, **12**, 931–934.
23. Doench, J.G., Fusi, N., Sullender, M., Hegde, M., Vaimberg, E.W., Donovan, K.F., Smith, I., Tothova, Z., Wilen, C., Orchard, R. *et al.* (2016) Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. *Nat Biotechnol*, **34**, 184–191.
24. Cursons, J., Pillman, K.A., Scheer, K.G., Gregory, P.A., Foroutan, M., Hediyeh-Zadeh, S., Toubia, J., Crampin, E.J., Goodall, G.J., Bracken, C.P. *et al.* (2018) Combinatorial Targeting by MicroRNAs Co-ordinates Post-transcriptional Control of EMT. *Cell Syst*, **7**, 77–91 e77.
25. Xuan, J.J., Sun, W.J., Lin, P.H., Zhou, K.R., Liu, S., Zheng, L.L., Qu, L.H. and Yang, J.H. (2018) RMBase v2.0: deciphering the map of RNA modifications from epitranscriptome sequencing data. *Nucleic Acids Res*, **46**, D327–D334.
26. Nair, A.S. and Sreenadhan, S.P. (2006) A coding measure scheme employing electron-ion interaction pseudopotential (EIIP). *Bioinformation*, **1**, 197–202.
27. Liu, Q., Chen, J., Wang, Y., Li, S., Jia, C., Song, J. and Li, F. (2020) DeepTorrent: a deep learning-based approach for predicting DNA N4-methylcytosine sites. *Brief Bioinform*. doi: 10.1093/bib/bbaa124.
28. Feng, S., Liang, Y., Du, W., Lv, W. and Li, Y. (2020) LncLocation: Efficient Subcellular Location Prediction of Long Non-Coding RNA-Based Multi-Source Heterogeneous Feature Fusion. *Int J Mol Sci*, **21**.
29. Dou, L., Li, X., Ding, H., Xu, L. and Xiang, H. (2020) Prediction of m5C Modifications in RNA Sequences by Combining Multiple Sequence Features. *Mol Ther Nucleic Acids*, **21**, 332–342.
30. Tang, Q., Nie, F., Kang, J. and Chen, W. (2020) ncPro-ML: An integrated computational tool for identifying non-coding RNA promoters in multiple species. *Comput Struct Biotechnol J*, **18**, 2445–2452.
31. Wei, L., Zhou, C., Chen, H., Song, J. and Su, R. (2018) ACPred-FL: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides. *Bioinformatics*, **34**, 4007–4016.
32. Wei, L., Tang, J. and Zou, Q. (2017) SkipCPP-Pred: an improved and promising sequence-

based predictor for predicting cell-penetrating peptides. *BMC Genomics*, **18**, 742.
33. Qiang, X., Chen, H., Ye, X., Su, R. and Wei, L. (2018) M6AMRFS: Robust Prediction of N6-Methyladenosine Sites With Sequence-Based Features in Multiple Species. *Front Genet*, **9**, 495.
34. Manavalan, B., Basith, S., Shin, T.H., Lee, D.Y., Wei, L. and Lee, G. (2019) 4mCpred-EL: An Ensemble Learning Framework for Identification of DNA N(4)-methylcytosine Sites in the Mouse Genome. *Cells*, **8**.
35. Wei, L., Su, R., Luan, S., Liao, Z., Manavalan, B., Zou, Q. and Shi, X. (2019) Iterative feature representations improve N4-methylcytosine site prediction. *Bioinformatics*, **35**, 4930–4937.
36. Pan, G., Jiang, L., Tang, J. and Guo, F. (2018) A Novel Computational Method for Detecting DNA Methylation Sites with DNA Sequence Information and Physicochemical Properties. *Int J Mol Sci*, **19**.
37. Zhou, Y., Zeng, P., Li, Y.H., Zhang, Z. and Cui, Q. (2016) SRAMP: prediction of mammalian N6-methyladenosine (m6A) sites based on sequence-derived features. *Nucleic Acids Res*, **44**, e91.
38. Gao, F. and Zhang, C.T. (2004) Comparison of various algorithms for recognizing short coding sequences of human genes. *Bioinformatics*, **20**, 673–681.
39. Liu, B., Liu, F., Fang, L., Wang, X. and Chou, K.C. (2015) repDNA: a Python package to generate various modes of feature vectors for DNA sequences by incorporating user-defined physicochemical properties and sequence-order effects. *Bioinformatics*, **31**, 1307–1309.
40. Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T. and Kanehisa, M. (2008) AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res*, **36**, D202–205.
41. Dong, Q., Zhou, S. and Guan, J. (2009) A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, **25**, 2655–2662.
42. Guo, Y., Yu, L., Wen, Z. and Li, M. (2008) Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. *Nucleic Acids Res*, **36**, 3025–3030.
43. Liu, B., Liu, F., Wang, X., Chen, J., Fang, L. and Chou, K.C. (2015) Pse-in-One: a web server for generating various modes of pseudo components of DNA, RNA, and protein sequences. *Nucleic Acids Res*, **43**, W65–71.
44. Chen, W., Feng, P.M., Lin, H. and Chou, K.C. (2013) iRSpot-PseDNC: identify recombination spots with pseudo dinucleotide composition. *Nucleic Acids Res*, **41**, e68.
45. Guo, S.H., Deng, E.Z., Xu, L.Q., Ding, H., Lin, H., Chen, W. and Chou, K.C. (2014) iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. *Bioinformatics*, **30**, 1522–1529.
46. Chen, W., Lei, T.Y., Jin, D.C., Lin, H. and Chou, K.C. (2014) PseKNC: a flexible web server for generating pseudo K-tuple nucleotide composition. *Anal Biochem*, **456**, 53–60.
47. Qiu, W.R., Xiao, X. and Chou, K.C. (2014) iRSpot-TNCPseAAC: identify recombination spots with trinucleotide composition and pseudo amino acid components. *Int J Mol Sci*, **15**, 1746–1766.
48. Bhasin, M. and Raghava, G.P. (2004) Classification of nuclear receptors based on amino acid composition and dipeptide composition. *J Biol Chem*, **279**, 23262–23266.
49. Chen, Z., Zhao, P., Li, F., Leier, A., Marquez-Lago, T.T., Wang, Y., Webb, G.I., Smith, A.I.,

Daly, R.J., Chou, K.C. *et al.* (2018) iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics*, **34**, 2499–2502.
50. Chen, Z., He, N., Huang, Y., Qin, W.T., Liu, X. and Li, L. (2018) Integration of A Deep Learning Classifier with A Random Forest Approach for Predicting Malonylation Sites. *Genomics Proteomics Bioinformatics*, **16**, 451–459.
51. Zhao, Y., He, N., Chen, Z. and Li, L. (2020) Identification of Protein Lysine Crotonylation Sites by a Deep Learning Framework With Convolutional Neural Networks. *IEEE Access*, **8**, 14244–14252.
52. Chen, K., Jiang, Y., Du, L. and Kurgan, L. (2009) Prediction of integral membrane protein type by collocated hydrophobic amino acid pairs. *J Comput Chem*, **30**, 163–172.
53. Chen, K., Kurgan, L. and Rahbari, M. (2007) Prediction of protein crystallization using collocation of amino acid pairs. *Biochem Biophys Res Commun*, **355**, 764–769.
54. Chen, K., Kurgan, L.A. and Ruan, J. (2007) Prediction of flexible/rigid regions from protein sequences using k-spaced amino acid pairs. *BMC Struct Biol*, **7**, 25.
55. Chen, K., Kurgan, L.A. and Ruan, J. (2008) Prediction of protein structural class using novel evolutionary collocation-based sequence representation. *J Comput Chem*, **29**, 1596–1604.
56. Saravanan, V. and Gautham, N. (2015) Harnessing Computational Biology for Exact Linear B-Cell Epitope Prediction: A Novel Amino Acid Composition-Based Feature Descriptor. *OMICS*, **19**, 648–658.
57. Chen, Z., Zhou, Y., Song, J. and Zhang, Z. (2013) hCKSAAP_UbSite: improved prediction of human ubiquitination sites by exploiting amino acid pattern and properties. *Biochim Biophys Acta*, **1834**, 1461–1467.
58. Chen, Z., Chen, Y.Z., Wang, X.F., Wang, C., Yan, R.X. and Zhang, Z. (2011) Prediction of ubiquitination sites by using the composition of k-spaced amino acid pairs. *PLoS One*, **6**, e22930.
59. Wang, D., Zeng, S., Xu, C., Qiu, W., Liang, Y., Joshi, T. and Xu, D. (2017) MusiteDeep: a deep-learning framework for general and kinase-specific phosphorylation site prediction. *Bioinformatics*, **33**, 3909–3916.
60. Wang, J.T.L., Ma, Q., Shasha, D. and Wu, C.H. (2001) New techniques for extracting features from protein sequences. **40**, 426–441.
61. White, G. and Seffens, W. (1998) Using a neural network to backtranslate amino acid sequences. *Electronic Journal of Biotechnology*. *Vol 1 Num 3*, **1**.
62. Lin, K., May, A.C. and Taylor, W.R. (2002) Amino acid encoding schemes from protein structure alignments: multi-dimensional vectors to describe residue types. *J Theor Biol*, **216**, 361–365.
63. Lee, T.Y., Lin, Z.Q., Hsieh, S.J., Bretana, N.A. and Lu, C.T. (2011) Exploiting maximal dependence decomposition to identify conserved motifs from a group of aligned signal sequences. *Bioinformatics*, **27**, 1780–1787.
64. Tung, C.W. and Ho, S.Y. (2008) Computational identification of ubiquitylation sites from protein sequences. *BMC Bioinformatics*, **9**, 310.
65. Sandberg, M., Eriksson, L., Jonsson, J., Sjostrom, M. and Wold, S. (1998) New chemical descriptors relevant for the design of biologically active peptides. A multivariate characterization of 87 amino acids. *J Med Chem*, **41**, 2481–2491.
66. Chen, Y.Z., Chen, Z., Gong, Y.A. and Ying, G. (2012) SUMOhydro: a novel method for the

prediction of sumoylation sites based on hydrophobic properties. *PLoS One*, **7**, e39195.
67. Lee, T.Y., Chen, S.A., Hung, H.Y. and Ou, Y.Y. (2011) Incorporating distant sequence features and radial basis function networks to identify ubiquitin conjugation sites. *PLoS One*, **6**, e17331.
68. Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *J Protein Chem*, **19**, 269–275.
69. Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, **27**, 451–477.
70. Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. *Am J Phys Anthropol*, **129**, 121–131.
71. Lin, Z. and Pan, X.M. (2001) Accurate prediction of protein secondary structural content. *J Protein Chem*, **20**, 217–220.
72. Dubchak, I., Muchnik, I., Holbrook, S.R. and Kim, S.H. (1995) Prediction of protein folding class using global description of amino acid sequence. *Proc Natl Acad Sci U S A*, **92**, 8700–8704.
73. Dubchak, I., Muchnik, I., Mayor, C., Dralyuk, I. and Kim, S.H. (1999) Recognition of a protein fold in the context of the Structural Classification of Proteins (SCOP) classification. *Proteins*, **35**, 401–407.
74. Cai, C.Z., Han, L.Y., Ji, Z.L., Chen, X. and Chen, Y.Z. (2003) SVM-Prot: Web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Res*, **31**, 3692–3697.
75. Cai, C.Z., Han, L.Y., Ji, Z.L. and Chen, Y.Z. (2004) Enzyme family classification by support vector machines. *Proteins*, **55**, 66–76.
76. Han, L.Y., Cai, C.Z., Lo, S.L., Chung, M.C. and Chen, Y.Z. (2004) Prediction of RNA-binding proteins from primary sequence by a support vector machine approach. *RNA*, **10**, 355–368.
77. Tomii, K. and Kanehisa, M. (1996) Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Eng*, **9**, 27–36.
78. Shen, J., Zhang, J., Luo, X., Zhu, W., Yu, K., Chen, K., Li, Y. and Jiang, H. (2007) Predicting protein-protein interactions based only on sequences information. *Proc Natl Acad Sci U S A*, **104**, 4337–4341.
79. Schneider, G. and Wrede, P. (1994) The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. *Biophys J*, **66**, 335–344.
80. Grantham, R. (1974) Amino acid difference formula to help explain protein evolution. *Science*, **185**, 862–864.
81. Chou, K.C. (2000) Prediction of protein subcellular locations by incorporating quasi-sequence-order effect. *Biochem Biophys Res Commun*, **278**, 477–483.
82. Chou, K.C. and Cai, Y.D. (2004) Prediction of protein subcellular locations by GO-FunD-PseAA predictor. *Biochem Biophys Res Commun*, **320**, 1236–1239.
83. Chou, K.C. (2005) Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics*, **21**, 10–19.
84. Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino acid

composition. *Proteins*, **43**, 246–255.
85. Chen, X., Qiu, J.D., Shi, S.P., Suo, S.B., Huang, S.Y. and Liang, R.P. (2013) Incorporating key position and amino acid residue features to identify general and species-specific Ubiquitin conjugation sites. *Bioinformatics*, **29**, 1614–1622.
86. Wang, J. and Wang, W. (1999) A computational approach to simplifying the protein folding alphabet. *Nat Struct Biol*, **6**, 1033–1038.
87. Zuo, Y., Li, Y., Chen, Y., Li, G., Yan, Z. and Yang, L. (2017) PseKRAAC: a flexible web server for generating pseudo K-tuple reduced amino acids composition. *Bioinformatics*, **33**, 122–124.
88. Liu, B., Fang, L., Wang, S., Wang, X., Li, H. and Chou, K.C. (2015) Identification of microRNA precursor with the degenerate K-tuple or Kmer strategy. *J Theor Biol*, **385**, 153–159.
89. Wang, R., Xu, Y. and Liu, B. (2016) Recombination spot identification Based on gapped k-mers. *Sci Rep*, **6**, 23934.
90. Liang, Y., Liu, S. and Zhang, S. (2015) Prediction of Protein Structural Classes for Low-Similarity Sequences Based on Consensus Sequence and Segmented PSSM. *Comput Math Methods Med*, **2015**, 370756.
91. Ogul, H. and Mumcuoglu, E.U. (2007) A discriminative method for remote homology detection based on n-peptide compositions with reduced amino acid alphabets. *Biosystems*, **87**, 75–81.
92. Kosiol, C., Goldman, N. and Buttimore, N.H. (2004) A new criterion and method for amino acid classification. *J Theor Biol*, **228**, 97–106.
93. Zuo, Y.C. and Li, Q.Z. (2010) Using K-minimum increment of diversity to predict secretory proteins of malaria parasite based on groupings of amino acids. *Amino Acids*, **38**, 859–867.
94. Melo, F. and Marti-Renom, M.A. (2006) Accuracy of sequence alignment and fold assessment using reduced amino acid alphabets. *Proteins*, **63**, 986–995.
95. Rakshit, S. and Ananthasuresh, G.K. (2008) An amino acid map of inter-residue contact energies using metric multi-dimensional scaling. *J Theor Biol*, **250**, 291–297.
96. Susko, E. and Roger, A.J. (2007) On reduced amino acid alphabets for phylogenetic inference. *Mol Biol Evol*, **24**, 2139–2150.
97. Liu, X., Liu, D., Qi, J. and Zheng, W.M. (2002) Simplified amino acid alphabets based on deviation of conditional probability from random background. *Phys Rev E Stat Nonlin Soft Matter Phys*, **66**, 021906.
98. Li, J. and Wang, W. (2007) Grouping of amino acids and recognition of protein structurally conserved regions by reduced alphabets of amino acids. *Sci China C Life Sci*, **50**, 392–402.
99. Peterson, E.L., Kondev, J., Theriot, J.A. and Phillips, R. (2009) Reduced amino acid alphabets exhibit an improved sensitivity and selectivity in fold assignment. *Bioinformatics*, **25**, 1356–1362.
100. Cannata, N., Toppo, S., Romualdi, C. and Valle, G. (2002) Simplifying amino acid alphabets by means of a branch and bound algorithm and substitution matrices. *Bioinformatics*, **18**, 1102–1108.
101. Li, T., Fan, K., Wang, J. and Wang, W. (2003) Reduction of protein sequence complexity by residue grouping. *Protein Eng*, **16**, 323–330.
102. Breiman, L. (2001) Random Forests. *Machine Learning*, **45**, 5–32.

103. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.-Y. (2017), *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Long Beach, California, USA, pp. 3149–3157.
104. Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine Learning*, **20**, 273–297.
105. Wang, S.C. (2003), *Interdisciplinary Computing in Java Programming*. Springer US, Boston, MA, pp. 3–15.
106. Altman, N.S. (1992) An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, **46**, 175–185.
107. Breiman, L. (1996) Bagging Predictors. *Machine Learning*, **24**, 123–140.