BioSeq-Analysis2.0: an updated stand-alone package for analyzing DNA, RNA, and protein sequences at sequence level and residue level based on machine learning approaches

Manual of stand-alone tool of BioSeq-Analysis2.0

2019-7-18

Home-page: http://bliulab.net/BioSeq-Analysis2.0/

Contents

1.BioSeq-Analysis-Res for residue-level analysis 2

1.1 Introduction 2

1.2 Installation 2

1.3 Function description 4

1.4 Commands 8

1.5 Methods description 20

2.BioSeq-Analysis-Seq for sequence-level analysis 21

2.1 Introduction 21

2.2 Installation 21

2.3 Function description 23

2.4 Commands 28

2.5 Methods description 52

Table 1-a. 7 residue-level modes for DNA sequences. 53

Table 1-b. 20 sequence-level modes for DNA sequences. 54

Table 2-a. 6 residue-level modes for RNA sequences. 54

Table 2-b. 14 sequence-level modes for RNA sequences. 55

Table 3-a. 13 residue-level modes for protein sequences 55

Table 3-b. 22 sequence-level modes for protein sequences. 56

Table 4. The names of the 148 physicochemical indices for dinucleotides. 57

Table 5. The names of the 12 physicochemical indices for trinucleotides. 58

Table 6. The names of the 90 physicochemical indices for dinucleotides. 58

Table 7. The names of the 6 physicochemical indices for dinucleotides. 59

Table 8. The names of the 22 physicochemical indices for dinucleotides. 59

Table 9. The names of the 11 physicochemical indices for dinucleotides. 59

Table 10. The names of the 547 physicochemical indices for amino acids. 59

Table 11. The names of the 3 physicochemical indices for amino acids. 63

Table 12. The names of the 2 physicochemical indices for amino acids. 63

References 63

1. BioSeq-Analysis-Res for residue-level analysis

1.1 Introduction

The platform **BioSeq-Analysis2.0** stand-alone package has two parts, for this section, we will introduce the residue-level analysis tool, for convenience, we call it BioSeq-Analysis-Res. The **BioSeq-Analysis-Res** is a updated platform for residue level analysis of DNA, RNA and Protein based on machine learning approaches, which can automatically implement the main procedures for constructing a predictor based on machine learning techniques, including feature extraction, parameter optimization, model training and performance evaluation. In the feature extraction step, totally 26 modes were provided for users, of which 7 for DNA residues, 6 for RNA residues and 13 for protein residues. In the predictor construction step, four machine learning algorithms are available: support vector machine (SVM) (1), random forest (RF) (2,3),conditional random fields(4). In order to handle large dataset, the stand-alone package of **BioSeq-Analysis-Res** is given. More details will be introduced in the following parts of the manual.

1.2 Installation

The **BioSeq-Analysis-Res** package can be run on Linux (64-bit) and Windows (64-bit) operating system. The full package and documents of **BioSeq-Analysis2.0** are available at http://bliulab.net/BioSeq-Analysis2.0/download.

For Windows

The Windows 7 or later versions are supported.
Before using **BioSeq-Analysis-Res**, the Python software should be first installed and configured. Python 2.7 64-bit is recommended, which can be downloaded from https://www.python.org.

The next step is the installation and configuration of LIBSVM (5), which you can download from (Version 3.22, December 2016)
https://www.csie.ntu.edu.tw/~cjlin/libsvm/#download

Then extract the package to BioSeq-Analysis-Res as a folder named libsvm. After un-zip the downloaded package, make sure that the "libsvm.dll" is available in the directory "..\libsvm\windows", and then put the file "__init__.py" and "checkdata.py" which is in the directory "**..\ supplement**" into the folder" ..\libsvm ". Next, put the "__init__.py" and "plotroc.py" which is in the ".. \ supplement" into the directory "..\libsvm\python".

The FlexCRFs(6)is also needed for **BioSeq-Analysis-Res,** so you can download it from:
http://flexcrfs.sourceforge.net/download.html.
Then extract the package to **BioSeq-Analysis-Res** as a folder named FlexCRFs-0.3, and you need makefile for FlexCRFs-0.3,
For more details you can see the flexcrf-manual in \FlexCRFs-0.3\docs\.

Then, the tool gnuplot (7) is need, which you can download from (Version4.6.5): https://sourceforge.net/projects/gnuplot/files/gnuplot/4.6.5/gp465-win32.zip/download

After installed the gnuplot, the Python package Numpy (8), SciPy (9), and matplotlib (10) should be downloaded from here: http://www.lfd.uci.edu/~gohlke/pythonlibs/, or use the following command to install :
> pip install numpy-<version>+mkl-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl
> pip install matplotlib-<version>-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl
> pip install matplotlib-<version>-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl

The Python package scikit-learn (11) should be downloaded and installed from: http://scikit-learn.org/dev/install.html, or use the following commands if Internet is accessible:
> pip install scikit-learn

The Python package imbalanced-learn (12) can be installed by using this command line:
> pip install -U imbalanced-learn

The Python package pandas (13) can be installed by using this command line:
> pip install pandas

For Linux

For Linux operating system, the libsvm and the flexcrfs should be configured as Windows firstly.

Extract the package to **BioSeq-Analysis-Res** as a folder named libsvm, then put the file "__init__.py" and "checkdata.py" which is in the directory "..\ supplement" into the folder" ..\libsvm ". Next, put the "__init__.py" and "plotroc.py" which is in the "..\ supplement" into the directory "..\libsvm\python".

Navigate to "~/usr/BioSeq-Analysis2.0/ BioSeq-Analysis-Res/libsvm" directory, and type the command:
> make
After executing successfully, then navigate to "~/usr/BioSeq-Analysis2.0/ BioSeq-Analysis-Res /libsvm/python" directory, and type the command:
> make

The FlexCRFs is also needed for **BioSeq-Analysis-Res,** so you can download it from: http://flexcrfs.sourceforge.net/download.html.
Then extract the package to **BioSeq-Analysis-Res** as **a folder named FlexCRFs**, and you need makefile for FlexCRFs,
Compile (go to FlexCRFs directory):
> make clean (remove any previous output)
> make all (compile FlexCRFs)
Install (you must login the system under the "root" privilege):
> make install (install FlexCRFs)
> make uninstall (uninstall FlexCRFs)

Given the root privilege
> sudo chmod –R 777 FlexCrs/
For more details you can see the flexcrf-manual in /FlexCRFs/docs.
If gnuplot has not been installed, use the following command lines to install gnuplot:
> sudo apt-get install gnuplot
Then, if your linux doesn't have scikit-learn, numpy, scipy, matplotlib and pandas, you should use the commods as follows:
> sudo apt-get install scikit-learn
> sudo apt-get install numpy
> sudo apt-get install scipy
> sudo apt-get install matplotlib
> sudo apt-get install pandas

Not Necessary Software

The predicted secondary structure features are generated by software PSIPRED (14) (15), which can be downloaded from
http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/.
The solvent accessible surface area features is generated by SPIDER2 (16,17), which can be downloaded from
http://sparks-lab.org/pmwiki/download/index.php?Download=yueyang/SPIDER2_local.tgz
The sequence conservation score features are generated by the package rate4site (18) (19), which can be installed by the following command:
> sudo apt-get install rate4site

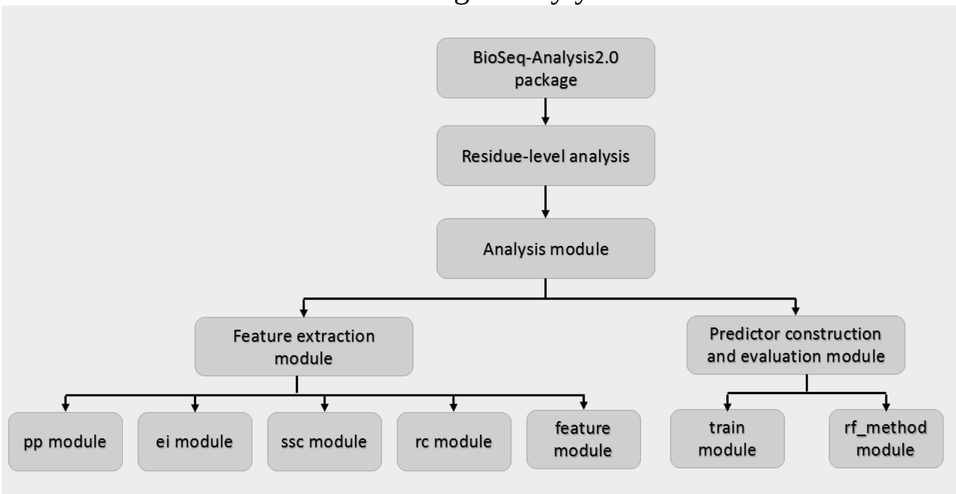
Now, **BioSeq-Analysis2.0** is ready to use.

1.3 Function description

1.3.1 Directory structure

The main directory contains several Python files and folders. "pp.py", "ei.py", "ssc_res.py", "rc.py", and "feature.py" are five executive Python scripts used for generating feature vectors based on the input sequence files and the selected feature extraction methods. "train.py" and "predict.py" are two executive scripts used for doing the analysis. "analysiss.py" is an executive Python scripts used for achieving the one-stop function. "ensemble.py" is used for ensemble learning based on the models generated by "train.py" or "analysiss.py". "optimization.py" is used for evaluating the performance of all the predictors generated by **BioSeq-Analysis-Res** so as to help the users to find the best predictor for a specific biological sequence analysis task. The details of their functions will be introduced in the following sections. "const.py" contains the constants used in the scripts. "util.py" provides the useful functions used in the scripts. "rf_method.py" contains the train methods of random forest. "rf_predict.py" contains the predict methods of random forest. In "data" folder, there are four subfolders: "example" folder contains the dataset files used in the example; "final_results" folder is used for storing the generated model file while the "gen_files" folder is used for storing the generated data files in the parameter selection process. The other files in the "data" folder are used for feature extraction methods. Modifications of these files are not suggested. "docs" folder contains the related documents of **BioSeq-Analysis-Res.**

"libsvm" folder contains the LIBSVM package. The tool for drawing ROC curve is in the "gnuplot" folder. "psiblast" folder contains the tools used for generating frequency profiles of protein sequences. To be noticed, the folder "libsvm", "gnuplot", "psiblast", you need download the software and configured by yourself.



The main module of the BioSeq-Analysis2.0 for residue-level analysis

1.3.2 Feature extraction

Scripts

"pp.py", "rc.py", "ssc.py", "ei.py"and "feature.py" There are six executive Python scripts used for generating feature vectors based on the input sequence files and the selected feature extraction methods.

The "rc.py" is used for calculating the modes in the sequence-based category and position-based category.The "pp.py" is used for calculating the modes in physicochemical property category. The "ei.py" is used for calculating the modes in the category profile-based. The "ssc.py" is used for calculating the modes in ml-based features category and predict rna secondary structure. The "feature.py" is used for calculating multiple modes in the four categories and achieving linear splicing for the feature vectors.

Input and output

The input file for "pp.py", "rc.py", "ssc.py", "ei.py" and "feature.py" should be a sequence file and a label file. The sequence file should be in a valid FASTA format that consists of a single initial line beginning with a greater-than symbol (">") in the first column, followed by lines of sequence data. The label file should be in a valid FASTA format that consists of a single initial line beginning with a greater-than symbol (">") in the first column, followed by lines of label data.

The words right after the ">" symbol in the single initial line are optional and only used for the purpose of identification and description.

For example, a valid FASTA format as follows:

Sequence Input:

```
>example
gacCagcttttaaaccgactccgtgctactgacgacca
```

Label Input:

```
>example
1 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0
```

The output file formats support three choices that are suitable for downstream computational analyses, such as machine learning. The first and the default choice is the tab format. In this format, all data is separated by TABs. The second one is the LIBSVM's sparse data format. For this format, each line contains an instance and is ended by a '\n' character, like <label> <index1>:<value1> <index2>:<value2> ... . The <label> is a category label of the residue. The pair <index>:<value> gives a feature (attribute) value: <index> is an integer starting from 1 and <value> is a real number. The third output format is the csv format. This format is similar to the tab format. The only difference is the separation characters between data are commas.

1.3.3 Classifier construction

The classifier construction part includes five main scripts: "train.py", "predict.py", "analysis.py", and "optimization.py".

train.py

Basic functions

The "train.py" is used for training predictors and evaluating their performance based on the input benchmark datasets. Both binary classification and multiclass classification are supported. There are three main processes of "train.py", including parameter selection, model training and cross validation. In the parameter selection process, the parameters of machine learning algorithm, SVM or RF are optimized on the validation sets. In this process, the multiprocessing technique is employed to significantly reduce the computational cost. In the model training process, SVM, RF, CRF is employed to train the prediction models. Finally, in the cross validation process, the performance of the constructed predictors is evaluated by k-fold cross-validation, jackknife or independent dataset test which can be selected by users. For more details of these three processes, please refer to the "**Methods description**" section.

Input and output

The input files of "train.py" are at least two files of feature vectors in LIBSVM format or CSV format generated by the feature extraction methods in"pp.py", "position.py", "profile_res.py", "mlss.py", "seq.py" and "feature.py". Two files need to be input, one is the sequence file, another is the label files. For binary classification problem, there are must two kind labels in the label files.For multiclass classification, at least three kind labels are needed. The output file is the trained SVM model or trained Random Forest model listing the parameters used in the training process and the log information, and the CRF method only can use through the analysisss.py, and the details you can see the analysiss.py. for example:

```
c,128,g,0.5,b,0,bi_or_multi,0
svm_type c_svc
kernel_type rbf
gamma 0.5
nr_class 2
total_sv 2871
rho 33.5904
label 1 -1
nr_sv 1441 1430
SV
128 1:0.00108139 2:0.00108139 3:0.00108139 ……
……
```

predict.py

Basic functions

The "predict.py" predicts the unseen samples independent from the benchmark dataset based on the trained model generated by using "train.py". For binary classification, the performance of the constructed predictors is evaluated by five common performance measures, and the corresponding ROC curves can also be generated. For multiclass classification, only one measure is calculated. For more information of these functions, please refer to the "**Methods description**" section.

Input and output

The input file of “predict.py” is an independent file of feature vectors in LIBSVM format or CSV format generated by feature extraction methods. If the label information of the samples is available, the performance measures of the predictors will be calculated based on the predicted labels and the input real labels, otherwise, the performance will not be evaluated. One label should be listed in each line in the label file, for example:

```
1
1
1
0
0
0
……
```

The output of “predict.py” is a file containing the predicted labels in the same format as the input label file.

analysis.py

Basic functions

The “analysiss.py” is the core executable file for the **BioSeq-Analysis-Res** standalone package. Its main role is training predictors and evaluating their performance based on the input benchmark datasets, and achieving parameter optimization at the same time. Both binary classification and multiclass classification are supported. There are five main processes of “analysiss.py”, including parameter selection, combination of the features, model training, cross validation and prediction on the independent dataset.

In process of the parameter selection, the parameters of feature extraction and machine learning are optimized on the validation sets. In this process, the multiprocessing technique is employed to significantly reduce the computational cost. In the process of combination of the features, the feature vectors will be achieved linear splicing. In the process of model training, the LIBSVM package, “rf_method.py” or FlexCRFs-0.3 package is employed to train the prediction models. Then, in the process of cross validation, the performance of the constructed predictors is evaluated by k-fold cross-validation, jackknife or independent dataset test which can be selected by users. Finally, in the process of prediction on the independent dataset, the unseen samples independent from the benchmark dataset will be predicted based on the trained model generated before. For binary classification, the performance of the constructed predictors is evaluated by five common performance measures, and the corresponding ROC curves can also be generated.

For multiclass classification, only one measure is calculated. For more details of these three processes, please refer to the “**Methods description**” section.

Input and output

The input files of “analysiss.py” are two files one file is biological sequence, another file is label sequence, which are in FASTA format. For binary classification problem, there are must two kind labels in the label files. For multiclass classification, at least three kind labels are needed. The output file contains the trained SVM model , Random Forest model or the CRF model listing the parameters used in the training process and the log information, for example:

```
c,128,g,0.5,b,0,bi_or_multi,0
svm_type c_svc
kernel_type rbf
gamma 0.5
```

```
nr_class 2
total_sv 2871
rho 33.5904
label 1 0
nr_sv 1441 1430
SV
128 1:0.00108139 2:0.00108139 3:0.00108139 ……
……
```

When there is an independent dataset, if the label information of the samples is available, the performance measures of the predictors will be calculated based on the predicted labels and the input real labels, otherwise, the performance will not be evaluated. One label should be listed in each line in the label file, for example:

```
1
1
1
0
0
0
……
```

If there has independent dataset, the output of "analysiss.py" will have a file containing the predicted labels in the same format as the input label file.

1.4 Commands

"rc.py" usage

Command line arguments for "rc.py":

Required	descript
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
method	The method name.
-labels	The input label file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-sp { under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

-fragment	If you use the fragment method, you need set the value ‘1’, or set ‘0’, default is 0.
-size	The size of sliding window. If you use the fragment method, the size don’t need set.

“pp.py” usage

Command line arguments for “pp.py”:

Required	descript
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
method	The method name.
-labels	The input label file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-sp { under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-fragment	If you use the fragment method, you should set the value ‘1’, or set ‘0’, default is 0.
-size	The size of sliding window. If you use the fragment method, the size don’t need set.

“ei.py” usage

Command line arguments for “ei.py”:

Required	descript
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
method	The method name.
-labels	The input label file in FASTA format.

Optional	description

-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-sp { under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-fragment	If you use the fragment method, you should set the value '1', or set '0', default is 0.
-size	The size of sliding window. If you use the fragment method， the size don't need set.

"ssc.py" usage

Command line arguments for "ssc.py":

Required	descript
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
method	The method name.
-labels	The input label file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-sp { under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-fragment	If you use the fragment method, you should set the value '1', or set '0'. default is 0.
-size	The size of sliding window. If you use the fragment method， the size don't need set.

“feature.py” usage

Command line arguments for “feature.py”:

Required	description
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
method	You can input several methods. The vector of each method implements linear merging. Up to 3 methods.
-labels	The input label file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).For Top-n-gram, PDT-Profile, DT, AC-PSSM, CC-PSSM, ACC-PSSM, PDT methods.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-sp {under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.
-fragment	If you use the fragment method, you should set the value ‘1’, or set ‘0’ .
-size	The size of sliding window. If you use the fragment method， the size don’t need set.

“train.py” usage

Command line arguments for “train.py”:

required	description
files	The input files. If the algorithm is set as SVM, the format of files should be LIBSVM format; if the algorithm is set as rf, the format of files should be csv format. For binary classification, two files needed. For multiclass classification, at least three files needed.
-m M	The name of the trained SVM model. Only for svm and rf.
-label_dict	Record each residue sequence’s label distribution.

Optional	description
-h, --help	Show this help message and exit.
-p {ACC,MCC,AUC}	The performance metric used for parameter selection. Default value is “ACC”.
-v V	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character “j”) jackknife cross validation.
-ind	The independent test dataset.
-ml {svm, rf}	The method of machine learning. svm is support vector machine; rf is random forest. (default is svm)
-opt	If the algorithm is set as svm: 0: small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. If the algorithm is set as rf: 0: small range set number of trees from 100 to 600, step is 200. 1: large range set number of trees from 100 to 600, step is 100. If the algorithm is set as oet_knn: 0: small range set neighbors from 1 to 30, step is 2. 1: large range set neighbors from 1 to 30, step is 1. Default value is 0.
-b {0,1}	Whether to train a SVC or SVR model for probability estimates, 0 or 1. Default value is 0.
-cpu	The maximum number of CPU cores used for multiprocessing during parameter selection process. Default value is 1.
-bp {1, 0}	The option of batch processing. 1 is run batch processing, 0 is not. Default is 0.

“predict.py” usage

Command line arguments for “predict.py”:

required	description
inputfiles	The input sequence files in LIBSVM format.
-m M	The name of the trained SVM model.

optional	description

-h, --help	Show this help message and exit.
-labels LABELS	The real label file. Optional.
-ml {svm, rf }	The method of machine learning. rf is Random Forest. (default is svm)
-o O	The output file name listing the predicted labels. The default name is “output_labels.txt”.

“analysis.py” usage

Command line arguments for “analysiss.py”:

Required	description
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
-model	The name of the trained model.
-method	The method names. You can input several methods. The vector of each method implements linear merging. Up to 3 methods.
-labels	The input label file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-b{0, 1}	Whether to train a SVC or SVR model for probability estimates, 0 or 1.(default=0). For svm method.
-v	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character “j”) jackknife cross validation.
-opt	Set the range of parameters to be optimized. 0: For svm, small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. For random forest, trees from 100 to 600, step is 200. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. For random forest, trees from 100 to 600, step is 100. (default=0).
-p {ACC,MCC,AUC}	The performance metric used for parameter selection. Default value is “ACC”.
-ind	The independent test dataset.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).For Top-n-gram, PDT-Profile, DT, AC-PSSM, CC-PSSM, ACC-PSSM, PDT methods and the number of CPU cores used for multiprocessing during parameter selection process.

-ml {svm, rf, crf}	The method of machine learning. rf is Random Forest. Oet_knn is Optimized Evidence-Theoretic K-Nearest Neighbor. Cda is covariance discriminant algorithm (default is svm)
-rl	The real label file. Optional.
-sp {under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.
-fragment	If you use the fragment method, you should set the value ‘1’, or set ‘0’.
-size	The size of sliding window. If you use the fragment method， the size don’t need set.

“optimization.py” usage

Command line arguments for “optimization.py”:

Required	description
inputfiles	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
-model	The name of the trained model.
-labels	The input sequence file in FASTA format.

Optional	description
-h, --help	Show this help message and exit.
-v	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character “j”) jackknife cross validation.
-opt	Set the range of parameters to be optimized. 0: For svm, small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. For random forest, trees from 100 to 600, step is 200. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. For random forest, trees from 100 to 600, step is 100. (default=0).
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).
-ml { svm, rf }	The method of machine learning. rf is Random Forest. (default is svm)

-sp { under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.
-fragment	If you use the fragment method, you should set the value '1', or set '0'. Default 0.
-size	The size of sliding window. If you use the fragment method， the size don't need set.

Example

Four examples of using **BioSeq-Analysis-Res** to construct machine learning predictor for solving a specific task in bioinformatics are given.

Example for residue level of DNA sequence.

Reconstructing the predictor iEnhancer-2L for identify enhancers based on the benchmark dataset(20) by using **BioSeq-Analysis-Res.**
The benchmark dataset contains 1484 positive samples and 1484 negative samples. The benchmark dataset are available at
http://bliulab.net/iEnhancer-EL/data/
In this example, the files "dna_frag_seq.txt" and "dna_frag_label.txt" contain the sequence dataset and label dataset of the benchmark dataset, respectively. All these two files are available in the "/data/example" folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/dna_frag_seq.txt DNA -method One-hot -ml svm
-labels ./data/example/dna_frag_label.txt -fragment 1 -model dna.model -opt 0 -v 5
-cpu 5
```

The output informations is as follows:

```
----------- Job is doing, please wait    -----------

Processing...
Parameters selecting of features done!

Combine the features of given methods and train it...
Method TPC is calculating...

The output file(s) can be found here:
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/final_results/
dna_frag_seq /Category~1_svm.txt

  /home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/final_results/
dna_frag_seq /Category~0_svm.txt
Processing on the best parameters...
Parameter selection is in processing...

Iteration    c =    -5    g =    -1    finished.
Iteration    c =    -5    g =    -4    finished.
Iteration    c =    -5    g =    -10    finished.
```

```
Iteration    c =    -5    g =    -7    finished.
Iteration    c =    -5    g =    2    finished.
Iteration    c =    -5    g =    5    finished.
Iteration    c =    -2    g =    -10    finished.
Iteration    c =    -2    g =    -7    finished.
Iteration    c =    -2    g =    -4    finished.
Iteration    c =    -2    g =    5    finished.
Iteration    c =    -2    g =    -1    finished.
Iteration    c =    -2    g =    2    finished.
Iteration    c =    1    g =    -10    finished.
Iteration    c =    1    g =    -7    finished.
Iteration    c =    1    g =    -4    finished.
Iteration    c =    1    g =    -1    finished.
Iteration    c =    1    g =    2    finished.
Iteration    c =    4    g =    -10    finished.
Iteration    c =    1    g =    5    finished.
Iteration    c =    4    g =    -7    finished.
Iteration    c =    4    g =    -4    finished.
Iteration    c =    4    g =    -1    finished.
Iteration    c =    4    g =    2    finished.
Iteration    c =    4    g =    5    finished.
Iteration    c =    7    g =    -10    finished.
Iteration    c =    7    g =    -7    finished.
Iteration    c =    7    g =    -4    finished.
Iteration    c =    7    g =    -1    finished.
Iteration    c =    7    g =    2    finished.
Iteration    c =    7    g =    5    finished.
Iteration    c =    10    g =    -10    finished.
Iteration    c =    10    g =    -7    finished.
Iteration    c =    10    g =    -4    finished.
Iteration    c =    10    g =    -1    finished.
Iteration    c =    10    g =    2    finished.
Iteration    c =    10    g =    5    finished.
The time cost for parameter selection is 218.26s
Parameter selection completed.

The optimal parameters for the dataset are: C =    1024    gamma =    0.0009765625

The cross validation results are as follows:
ACC = 0.7369
MCC = 0.4783
AUC = 0.8126
Sn    = 0.6716
Sp    = 0.8020

The ROC curve has been saved. You can check it here:
./data/ final_results/cv_roc.png

Model training completed.
The model has been saved. You can check it here:
./data/ final_results/dna .model

Done.
Used time: 277.22s
Total used time: 289.60s
```

The generated ROC curve is shown in **Fig. 1**.

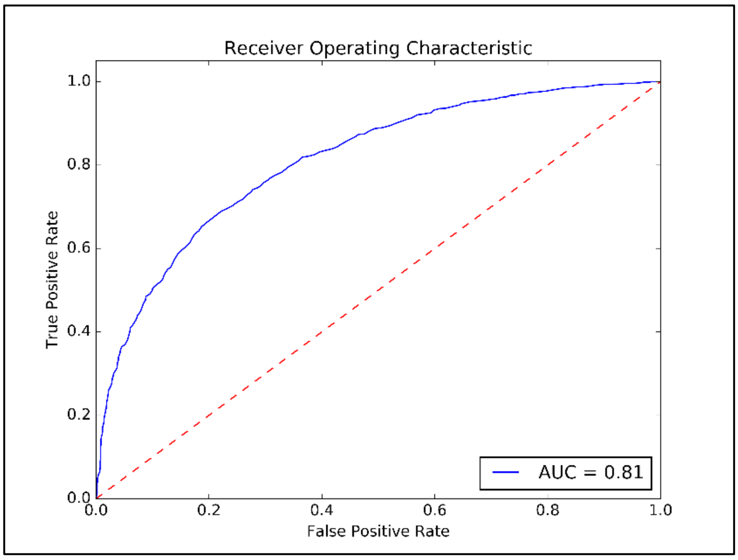


Fig .1. The ROC curve of cross validation

As shown in this example, the iEhancer-2L/iEnhancer-EL can be easily constructed based on the benchmark dataset by using the script "analysis.py".

Example for residue level of RNA sequence.

N6-Methyladenosine (m6A) is an RNA methylation modification at the nitrogen-6 position of the adenosine base(21). Reconstructing the predictor for identification m6A precursors based on the benchmark dataset (22) by using BioSeq-Analysis-Res.
The benchmark dataset contains 1452 positive samples and 1348 negative samples.
All these two files are available in the "/data/example" folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/rna_frag_seq.txt RNA -method DPC -ml rf
-labels ./data/example/rna_frag_label.txt -fragment 1 -model rna.model -opt 0 -v 5
-cpu 5
```

The output informations is as follows:

```
------------ Job is doing, please wait    ------------
Processing...
Parameters selecting of features done!

Combine the features of given methods and train it...
Method DPC is calculating...
The output file(s) can be found here:
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/ final_results/
rna_frag_seq /Category~0_csv.txt
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/ final_results/
rna_frag_seq /Category~1_csv.txt
Processing...
Parameter selection is in processing...
Trees are 100...
Trees are 300...
```

```
Trees are 500...

The time cost for parameter selection is 74.29s
Parameter selection completed.

The optimal parameter for the dataset is: Parameter = 500

Model training is in processing...
The cross validation results are as follows:
ACC = 0.6868
MCC = 0.3728
AUC = 0.7387
Sn   = 0.7073
Sp   = 0.6647

The ROC curve has been saved. You can check it here:
./data/ final_results/cv_roc.png
Model training completed.
The model has been saved. You can check it here:
./data/ final_results/ rna .model

Total used time: 6186.99s
```

The generated ROC curve is shown in **Fig. 2**.

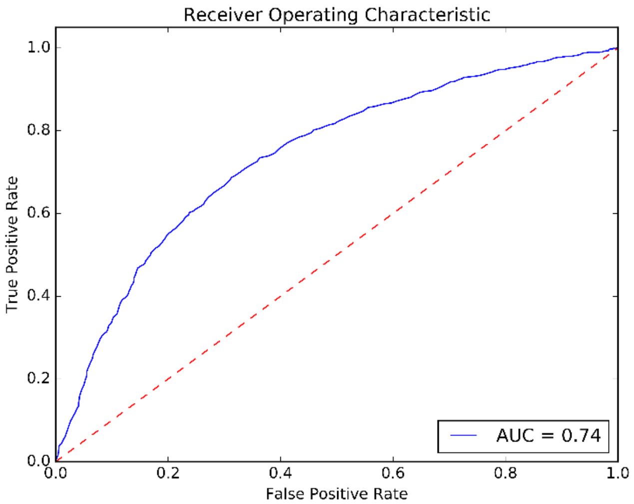


Fig .2. The ROC curve of cross validation

As shown in this example, the m6A identification predictors can be easily constructed based on the benchmark dataset by using the script "analysis.py".

Example of protein

Reconstructing the predictor for Protein disordered region identification based on the benchmark dataset(23), by using **BioSeq-Analysis2.0.**
The benchmark dataset contains 5442 positive samples and 10232 negative samples..

In this example, the files "protein_seq.txt" and "protein_label.txt" contain the sequence dataset and label dataset of the benchmark dataset, respectively. T All these files are available in the "/data/example" folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/ protein_seq.txt Protein
-labels ./data/example/protein_label -method One-hot-6bit -ml crf -model
protein.model -size 13 -opt 0 -v 5 -cpu 5
```

The output informations is as follows:

```
------------ Job is doing, please wait    ------------

there are 2 kinds
Processing...
Parameters selecting of features done!

Combine the features of given methods and train it...
Method PSFM is calculating...
The output file(s) can be found here:
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/final_results/protein
_seq/Category~1_svm.txt
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/final_results/protein
_seq/Category~0_svm.txt
Processing on the best parameters...
This is model: ./ data/final_results/protein_seq/crf_app/Fold1/model.txt
This is model:    ./ data/final_results/protein_seq/crf_app/Fold2/model.txt This is
model: ./ data/final_results/protein_seq/crf_app/Fold3/model.txt
This is model: ./ data/final_results/protein_seq/crf_app/Fold4/model.txt    This is
model: ./ data/final_results/protein_seq/crf_app/Fold5/model.txt

ACC = 0.7246
MCC = 0.3640
AUC = 0.7472
Sn = 0.4875
Sp = 0.8507
The ROC curve has been saved. You can check it here:
/home/First_project/BioSeq-Analysis2.0/BioSeq-Analysis-Res/data/final_results/
protein_seq /cv_roc.png

Done.
```

The generated ROC curve is shown in **Fig. 3**.

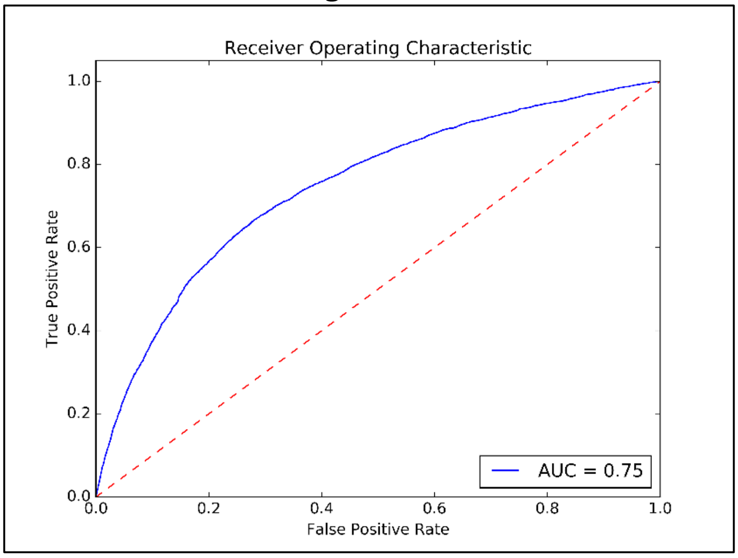


Fig .3. The ROC curve of cross validation

As shown in this example, the predictor can be easily constructed based on the benchmark dataset by using the script “analysis.py”.

1.5 Methods description

1.5.1 Feature extraction

The **BioSeq-Analysis-Res** stand-alone package is able to generate totally 26 different modes of pseudo components for Deoxyribonucleic acid, Ribonucleic acid, and Amino acid, including 7 modes for Deoxyribonucleic acid (**Table 1-a**), 6 modes for Ribonucleic acid (**Table 2-a**), and 14 modes for Amino acid (**Table 3-a**). The detailed information and reference of the 26 methods will be introduced in BioSeq-Analysis-Res description document which can be downloaded from here:
http://bliulab.net/BioSeq-Analysis2.0/doc/.
For many biological residue analysis tasks, the training sets are imbalanced. As a result, a predictor trained by a skewed dataset would inevitably lead to a bias consequence (24). The undersampling is widely used to minimize this bias consequence. For undersampling, some samples are randomly removed from the large class to make the number of samples in different classes the same. In **BioSeq-Analysis2.0**, the SMOTE algorithm (25) were employed to generate the hypothetical samples for this purpose.

1.5.2 Parameter selection

In LIBSVM there are two parameters *c* and *g* which can determine the performance of the predictor. In Random Forest there is one parameter *t* which can determine the performance of the predictor. **BioSeq-Analysis-Res** is able to automatically optimize these parameters based on the best performance on the validation set. Users can choose a range of the parameters for optimizing. For more information of the input format, please refer to “**Commands**” section.
To improve the efficiency of this procedure, multiprocessing technique is applied, which significantly reduces the computational cost. One of the three performance measures, including Accuracy (ACC), Mathew’s Correlation Coefficient (MCC) and Area Under roc Curve (AUC) can be used as the golden standard to optimize the parameters.

1.5.3 Predictor construction

In the model training process, this model is trained based on LIBSVM with RBF kernel, Random Forest, and a sequence labeling model—CRF.

1.5.4 Cross validation

BioSeq-Analysis-Res provides three types of cross validation options, including k-fold cross validation, jackknife (leave-one-out cross validation) and independent dataset test, which can be chosen by the argument “-v”. Please refer to “**Commands**” section for more details.
For binary classification, the performance of the predictor is measured by five common performance measures, including the accuracy (ACC), Mathew’s Correlation Coefficient (MCC), Area Under roc Curve (AUC), sensitivity (Sn), and specificity (Sp). Furthermore, the ROC (Receiver Operating Characteristic) (26) curve will also be generated and saved in a PNG file.
For multiclass classification, only the performance measure of ACC is calculated since the other measures are not suitable for multiclass classification.
Besides, if the parameter “-b” of libsvm is set or using the random forest, the prediction probability values will be output and save as a file, thus users can do further analysis with these data.

1.5.5 Residue prediction

The “predict.py” is used to predict the unseen samples based on the model trained by using “train.py”. The performance of the predictors can be further evaluated on the independent datasets. If the label information of the independent dataset is not available, the performance of the predictor will not be evaluated, and only the predicted labels are given. Otherwise, this script will output the predicted labels. For binary classification, the five performance measures (ACC, MCC, AUC, Sn, and Sp) will be calculated along with the corresponding ROC curve saved as a PNG file; for multiclass classification, only the performance measure ACC will be calculated.

2. BioSeq-Analysis-Seq for sequence-level analysis

2.1 Introduction

The platform **BioSeq-Analysis2.0** stand-alone package has two parts. For this section, we will introduce the sequence-level analysis tool, for convenience, we call it **BioSeq-Analysis-Seq**. The **BioSeq-Analysis-Seq** is a package for DNA, RNA and protein sequence analysis based on machine learning approaches, which can automatically implement the main procedures for constructing a predictor based on machine learning techniques, including feature extraction, parameter optimization, model training and performance evaluation. In the feature extraction step, totally 56 modes were provided for users, of which 20 for DNA sequences, 14 for RNA sequences and 22 for protein sequences. In the predictor construction step, four machine learning algorithms are available: support vector machine (SVM) (1), random forest (RF) (2,3), Optimized Evidence-Theoretic K-Nearest Neighbor (OET-KNN) (27), and covariance discriminant algorithm (28). In order to handle large dataset, the stand-alone package of **BioSeq-Analysis-Seq** is given. More details will be introduced in the following parts of the manual.

2.2 Installation

The **BioSeq-Analysis-Seq** package can be run on Linux (64-bit) and Windows (64-bit) operating system. The full package and documents of **BioSeq-Analysis-Seq** are available at http://bliulab.net/BioSeq-Analysis2.0/download.

For Windows

The Windows 7 or later versions are supported.
Before using **BioSeq-Analysis-Seq**, the Python software should be first installed and configured. Python 2.7 64-bit is recommended, which can be downloaded from https://www.python.org.

The next step is the installation and configuration of LIBSVM (5), which you can download from (Version 3.22, December 2016) https://www.csie.ntu.edu.tw/~cjlin/libsvm/#download

Then extract the package to BioSeq-Analysis-Seq as a folder named libsvm. After un-zip the downloaded package, make sure that the “libsvm.dll” is available in the directory “..\libsvm\windows”, and then put the file “__init__.py” and “checkdata.py” which is in the directory “..\ supplement” into the folder“ ..\libsvm ”. Next, put the "__init__.py" and "plotroc.py" which is in the ".. \ supplement" into the directory "..\libsvm\python".

Then, the tool gnuplot (7) is need, which you can download from (Version4.6.5):
https://sourceforge.net/projects/gnuplot/files/gnuplot/4.6.5/gp465-win32.zip/download

After installed the gnuplot, the Python package Numpy (8), SciPy (9), and matplotlib (10) should be downloaded from here: http://www.lfd.uci.edu/~gohlke/pythonlibs/, or use the following command to install :
> pip install numpy-<version>+mkl-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl
> pip install matplotlib-<version>-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl
> pip install matplotlib-<version>-cp<ver-spec>-cp<ver-spec>m-<cpu-build>.whl

The Python package scikit-learn (11) should be downloaded and installed from: http://scikit-learn.org/dev/install.html, or use the following commands if Internet is accessible:
> pip install scikit-learn

The Python package imbalanced-learn (12) can be installed by using this command line:
> pip install -U imbalanced-learn

The Python package pandas (13) can be installed by using this command line:
> pip install pandas

For Linux

For Linux operating system, the libsvm should be configured as Windows firstly.

Extract the package to BioSeq-Analysis-Seq as a folder named libsvm, then put the file "__init__.py" and "checkdata.py" which is in the directory "..\ supplement" into the folder" ..\libsvm ". Next, put the "__init__.py" and "plotroc.py" which is in the ".. \ supplement" into the directory "..\libsvm\python".

Navigate to "~/usr/BioSeq-Analysis2.0/BioSeq-Analysis-Seq/libsvm" directory, and type the command:
> make
After executing successfully, then navigate to "~/usr/ BioSeq-Analysis2.0/BioSeq-Analysis-Seq/libsvm/python" directory, and type the command:
> make
If gnuplot has not been installed, use the following command lines to install gnuplot:
> sudo apt-get install gnuplot
Then, if your linux doesn't have scikit-learn,
numpy, scipy, matplotlib and pandas, you should use the commods as follows:
> sudo apt-get install scikit-learn
> sudo apt-get install numpy
> sudo apt-get install scipy
> sudo apt-get install matplotlib
> sudo apt-get install pandas

Not Necessary Software

The predicted secondary structure features are generated by software PSIPRED (14) (15), which can be downloaded from http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/.

The solvent accessible surface area features is generated by SPIDER2 (16,17), which can be downloaded from http://sparks-lab.org/pmwiki/download/index.php?Download=yueyang/SPIDER2_local.tgz

The sequence conservation score features are generated by the package rate4site (18) (19), which can be installed by the following command:

> sudo apt-get install rate4site

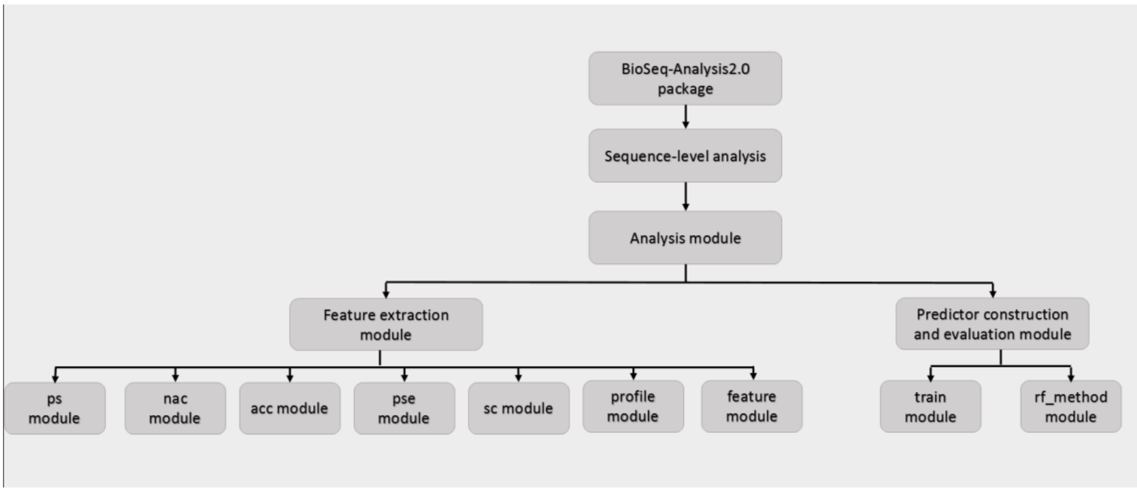
Now, **BioSeq-Analysis-Seq** is ready to use.

2.3 Function description

2.3.1 Directory structure

The main directory contains several Python files and folders. “nac.py”, “acc.py”, “pse.py”, “sc.py”, “profile.py”, “ps.py” and “feature.py” are seven executive Python scripts used for generating feature vectors based on the input sequence files and the selected feature extraction methods. “train.py” and “predict.py” are two executive scripts used for doing the analysis. “analysiss.py” is an executive Python scripts used for achieving the one-stop function. “ensemble.py” is used for ensemble learning based on the models generated by “train.py” or “analysiss.py”. “optimization.py” is used for evaluating the performance of all the predictors generated by **BioSeq-Analysis-Seq** so as to help the users to find the best predictor for a specific biological sequence analysis task. The details of their functions will be introduced in the following sections. “const.py” contains the constants used in the scripts. “util.py” provides the useful functions used in the scripts and “util_sc.py” provides some specific functions used for “sc.py”. “rf_method.py” contains the train methods of random forest. “rf_predict.py” contains the predict methods of random forest. “acc_pssm” folder contains the tools used for ACC-PSSM, AC-PSSM and CC-PSSM methods. “pdt” folder contains the tools used for PDT and PDT-Profile methods. “docs” folder contains the related documents of BioSeq-Analysis-Seq. In “data” folder, there are four subfolders: “example” folder contains the dataset files used in the example; “final_results” folder is used for storing the generated model file while the “gen_files” folder is used for storing the generated data files in the parameter selection process. The other files in the “data” folder are used for feature extraction methods. Modifications of these files are not suggested.

“libsvm” folder contains the LIBSVM package. The tool for drawing ROC curve is in the “gnuplot” folder. “psiblast” folder contains the tools used for generating frequency profiles of protein sequences. **These three folders are created by the users.**



The main module of the BioSeq-Analysis2.0 for sequence-level analysis

2.3.2 Feature extraction

Scripts

“nac.py”, “acc.py”, “pse.py”, “sc.py”, “profile.py”, “ps.py” and “feature.py”. There are seven executive Python scripts used for generating feature vectors based on the input sequence files and the selected feature extraction methods.

The “nac.py” is used for calculating the modes in the category nucleic acid composition or amino acid composition; the “acc.py” is used for calculating the modes in autocorrelation category. The “pse.py” is used for calculating the modes in the category pseudo nucleotide composition or pseudo amino acid composition. The “sc.py” is used for calculating the modes in predicted structure composition category. The “profile.py” is used for calculating the modes in profile-based features category. The “ps.py” is used for calculating the modes in predicted structure features category. The “feature.py” is used for calculating multiple modes in the six categories and achieving linear splicing for the feature vectors.

Input and output

The input file for “nac.py”, “acc.py”, “pse.py”, “profile.py”, “ps.py” and “feature.py” should be in a valid FASTA format that consists of a single initial line beginning with a greater-than symbol (“>”) in the first column, followed by lines of sequence data. The words right after the “>” symbol in the single initial line are optional and only used for the purpose of identification and description. For “sc.py”, the input file should be in a valid FASTA format with the secondary structure as follows:

```
>example
GCAUCCGGGUUGAGGUAGUAGGUUGUAUGGUUUAGAGUUACACCCUGGG
AGUUAACUGUACAACCUUCUAGCUUUCCUUGGAGC
((.((((((..(((.(((.((((((((((((..((.(..((...))..).))))))))))))))).))).)))..)))))))) (-31.60)
```

For “feature.py”, the input file should be in a valid FASTA format if the methods used in “sc.py”, and if the methods used in “nac.py”, “acc.py”, “pse.py”, “profile.py” or “ps.py”, the input file should be in a valid FASTA format with the secondary structure.

The output file formats support three choices that are suitable for downstream computational analyses, such as machine learning. The first and the default choice is the tab format. In this format, all data is separated by TABs. The second one is the LIBSVM’s sparse data format. For this format, each line contains an instance and is ended by a '\n' character, like <label> <index1>:<value1> <index2>:<value2> ... . The <label> is a category label of the sequence. The pair <index>:<value> gives a feature (attribute) value: <index> is an integer starting from 1 and <value> is a real number. The third output format is the csv format. This format is similar to the tab format. The only difference is the separation characters between data are commas.

Physicochemical Properties Selection

The Physicochemical Properties Selection file is a text file that contains a list of property names used for generating the modes in categories: autocorrelation, pseudo nucleotide composition/ pseudo amino acid composition. For example, if you want to use the “Rise”, “Tilt” and “Shift” of DNA dinucleotide for calculating, the Physicochemical Properties Selection file should be written as follows:

```
Rise
Tilt
Shift
```

After saving this file as “propChosen.txt” and specifying it using the command “-i propChosen.txt”, or just “I propChosen.txt”, the above three properties will be used in calculations. Meanwhile, you can also use the command “-a True” to select all the built-in physicochemical properties for the corresponding sequence type, which can be selected by using parameter DNA, RNA or PROTEIN.
The complete lists of physicochemical properties for DNA, RNA and protein sequences used in the stand-alone program are provided in **Table 4-12**.

User-defined Physicochemical Properties

In the user-defined physicochemical index files, each index should be represented in three lines. The first line must start with a greater-than symbol (">") in the first column. The words right after the ">" symbol in the single initial line are optional and only used for the purpose of identification and description of the index. The second line lists the names of the sequence compositions (i.e. amino acids, nucleotides, dinucleotides, or trinucleotides, etc), which should be sorted in the alphabet order, such as 'A' 'C' ... 'AA' 'AC'. All the elements in this line should be separated by TAB. The corresponding values of these sequence compositions are listed in the third line, which are separated by TAB.
For example, if you defined a physicochemical property “user_property”, the user-defined physicochemical index file should be written as follows:

```
> user_property
A     C     …     AA AC …
0.21        0.12        …     0.37     0.15        …
```

After saving this file as “user_defined.txt” and specifying it using the command “-e user_defined.txt”, or just “E user_defined.txt”, the properties defined by user will be used in calculations.

2.3.3 Classifier construction

The classifier construction part includes five main scripts: “train.py”, “predict.py”, “analysis.py”, “ensemble.py” and “optimization.py”.

train.py

Basic functions

The “train.py” is used for training predictors and evaluating their performance based on the input benchmark datasets. Both binary classification and multiclass classification are supported. There are three main processes of “train.py”, including parameter selection, model training and cross validation. In the parameter selection process, the parameters of machine learning algorithm, SVM or RF are optimized on the validation sets. In this process, the multiprocessing technique is employed to significantly reduce the computational cost. In the model training process, SVM or RF is employed to train the prediction models. Finally, in the cross validation process, the performance of the constructed predictors is evaluated by k-fold cross-validation, jackknife or independent dataset test which can be selected by users. For more details of these three processes, please refer to the “**Methods description**” section.

Input and output

The input files of “train.py” are at least two files of feature vectors in LIBSVM format or CSV format generated by the feature extraction methods in “nac.py”, “acc.py”, “pse.py” , “sc.py” and “feature.py”. For binary classification problem, two files need to be input, storing the positive samples and the negative samples, respectively. For multiclass classification, at least three files are needed. The output file is the trained SVM model or trained Random Forest model listing the parameters used in the training process and the log information, for example:

```
c,128,g,0.5,b,0,bi_or_multi,0
svm_type c_svc
kernel_type rbf
gamma 0.5
nr_class 2
total_sv 2871
rho 33.5904
label 1 -1
nr_sv 1441 1430
SV
128 1:0.00108139 2:0.00108139 3:0.00108139 ……
……
```

predict.py

Basic functions

The “predict.py” predicts the unseen samples independent from the benchmark dataset based on the trained model generated by using “train.py”. For binary classification, the performance of the constructed predictors is evaluated by five common performance measures, and the corresponding ROC curves can also be generated. For multiclass classification, only one measure is calculated. For more information of these functions, please refer to the “**Methods description**” section.

Input and output

The input file of “predict.py” is an independent file of feature vectors in LIBSVM format or CSV format generated by feature extraction methods. If the label information of the samples is available, the performance measures of the predictors will be calculated based on the predicted labels and the input real labels, otherwise, the performance will not be evaluated. One label should be listed in each line in the label file, for example:

```
+1
+1
```

```
+1
-1
-1
-1
……
```

The output of "predict.py" is a file containing the predicted labels in the same format as the input label file.

analysis.py

Basic functions

The "analysiss.py" is the core executable file for the BioSeq-Analysis-Seq standalone package. Its main role is training predictors and evaluating their performance based on the input benchmark datasets, and achieving parameter optimization at the same time. Both binary classification and multiclass classification are supported. There are five main processes of "analysiss.py", including parameter selection, combination of the features, model training, cross validation and prediction on the independent dataset.
In process of the parameter selection, the parameters of feature extraction and machine learning are optimized on the validation sets. In this process, the multiprocessing technique is employed to significantly reduce the computational cost. In the process of combination of the features, the feature vectors will be achieved linear splicing. In the process of model training, the LIBSVM package or "rf_method.py" is employed to train the prediction models. Then, in the process of cross validation, the performance of the constructed predictors is evaluated by k-fold cross-validation, jackknife or independent dataset test which can be selected by users. Finally, in the process of prediction on the independent dataset, the unseen samples independent from the benchmark dataset will be predicted based on the trained model generated before. For binary classification, the performance of the constructed predictors is evaluated by five common performance measures, and the corresponding ROC curves can also be generated.
For multiclass classification, only one measure is calculated. For more details of these three processes, please refer to the "**Methods description**" section.

Input and output

The input files of "analysiss.py" are at least two files of biological sequence in FASTA format. For binary classification problem, two files need to be input, storing the positive samples and the negative samples, respectively. For multiclass classification, at least three files are needed. The output file contains the trained SVM model or the Random Forest model listing the parameters used in the training process and the log information, for example:

```
c,128,g,0.5,b,0,bi_or_multi,0
svm_type c_svc
kernel_type rbf
gamma 0.5
nr_class 2
total_sv 2871
rho 33.5904
label 1 -1
nr_sv 1441 1430
SV
128 1:0.00108139 2:0.00108139 3:0.00108139 ……
……
```

When there is an independent dataset, if the label information of the samples is available, the performance measures of the predictors will be calculated based on the predicted labels and the input real labels, otherwise, the performance will not be evaluated. One label should be listed in each line in the label file, for example:

```
+1
+1
+1
-1
-1
-1
……
```

If there has independent dataset, the output of "analysiss.py" will have a file containing the predicted labels in the same format as the input label file.

ensemble.py

Basic functions

The "ensemble.py" is used for ensemble learning based on the models generated by "train.py" or "analysiss.py". Both binary classification and multiclass classification are supported. The weight of every model can be specified by users. Default values are 1.0. The strategy of prediction is weighted voting.

Input and output

The input file should be in tab format which can be generated by the scripts for feature extraction. The format of label file should be the same as that of "predict.py". The input model files are those generated by "train.py" or "analysis.py". For binary classification, four measures, including the accuracy (ACC), Mathew's Correlation Coefficient (MCC), sensitivity (Sn), and specificity (Sp) are used for performance evaluation. For multiclass classification, only ACC is calculated. The values of the measures will be printed on the screen.

optimization.py

Basic functions

The "ensemble.py" is used for batch processing. This scrip is used for evaluating the performance of all the predictors generated by **BioSeq-Analysis-Seq** so as to help the users to find the best predictor for a specific biological sequence analysis task.

Input and output

The input file should be in fasta format. The parameters are similar with those in "analysiss.py".

2.4 Commands

"nac.py" usage

Command line arguments for "nac.py":

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
method	The method name.

Optional	description

-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the sa me as that of input files.
-k K	The k value of kmer.
-m M	For mismatch. The max value inexact matching. (m<k). (default = 1)
-delta	For subsequence method. The value of penalized factor. (0<=delta<=1). (default = 1)
-r {0,1}	Whether consider the reverse complement or not. 1 means True, 0 means False. (default = 0)
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument "-f " is set as "svm", this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-ps	The input positive source file in FASTA format for IDKmer. Only for IDKmer method.
-ns	The input negative source file in FASTA format for IDKmer. Only for IDKmer method.
-max_dis	The max distance value of DR and Distance Pair. Only for DR and Distance Pair methods(default = 3).
-cp	The reduced alphabet scheme. Choose one of the four: cp_13, cp_14, cp_19, cp_20. Only for Distance Pair method.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

"acc.py" usage

Command line arguments for "acc.py":

Required	**description**
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
method	The method name.

Optional	**Description**
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-lag LAG	The value of lag.
-i I	The index file user chosen.

-e E	The user-defined index file.
-all_index	Choose all physicochemical indices.
-no_all_index	Do not choose all physicochemical indices, default.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument "-f " is set as "svm", this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-lamada	The value of lamada. Only for MAC, GAC, NMBAC methods (default=1).
-oli	Choose one kind of Oligonucleotide: 0 represents dinucleotide, default; 1 represents trinucleotide.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

"pse.py" usage

Command line arguments for "pse.py":

Required	**description**
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
method	The method name.

Optional	**description**
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-lamada	The value of lamada (default=2).
-w W	The value of weight (default=0.1).
-k K	The value of kmer, it works only with PseKNC method.
-e E	The user-defined index file, this parameter only needs to be set for PC-PseDNC-General, PC-PseTNC-General, SC-PseDNC-General, SC-PseTNC-General, PC-PseAAC-General or SC-PseAAC-General.
-all_index	Choose all physicochemical indices.
-no_all_index	Do not choose all physicochemical indices, default.

-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument “-f ” is set as “svm”, this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

“sc.py” usage

Command line arguments for “sc.py”:

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
method	The method name.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-k K	The number of k adjacent structure statuses (default=2). It works only with PseSSC method.
-n N	The maximum distance between structure statuses (default=0). It works only with PseDPC method.
-r R	The value of lambda, represents the highest counted rank (or tier) of the structural correlation along a RNA chain (default=2).
-w W	The weight factor used to adjust the effect of the correlation factors (default=0.1).
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument “-f ” is set as “svm”, this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.

-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

“profile.py” usage

Command line arguments for “profile.py”:

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
method	The method name.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-n N	For Top-n-gram, PDT-Profile methods. The value of top-n-gram. The value cam only be 1, 2 or 3.
-lamada	For PDT, PDT-Profile methods. The value of lamada
-max_dis	For DT methods. The max distance value of residues (default = 3).
-lag LAG	For ACC-PSSM, AC-PSSM and CC-PSSM methods. The value of lag (default = 2).
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument “-f ” is set as “svm”, this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. Default value is 1.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.

“ps.py” usage

Command line arguments for “ps.py”:

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
method	The method name.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument "-f " is set as "svm", this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. Default value is 1.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling for the datasets. Under is under sampling for the datasets.

"feature.py" usage

Command line arguments for "feature.py":

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
-method	The method names. You can input several methods. The vector of each method implements linear merging. Up to 3 methods.

Optional	description
-h, --help	Show this help message and exit.
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-k K	The number of k adjacent structure statuses. (default=2). It works with PseKNC, PseSSC, Kmer, RevKmer, IDKmer, Mismatch, Subsequence methods. If there are several methods, enter the values in turn.
-m M	For Mismatch. The max value inexact matching. (m<k) (default=1). If there are several methods, enter the values in turn.
-delta	For subsequence method. The value of penalized factor. (0<=delta<=1) (default=1). If there are several methods, enter the values in turn.

-r	Whether consider the reverse complement or not. 1 means True, 0 means False. For RevKmer methods. (default=0). Or the value of lambda, represents the highest counted rank (or tier) of the structural correlation along a RNA chain. For Triplet, PseSSC, PseDPC methods. (default=2). If there are several methods, enter the values in turn.
-oli	Choose one kind of Oligonucleotide: 0 represents dinucleotide, default; 1 represents trinucleotide. For DAC, DCC, DACC, TAC, TCC, TACC, MAC, GAC, NMBAC, AC, CC, ACC methods. If there are several methods, enter the values in turn.
-lamada	The value of lamada. For PseDNC, PseKNC, PC-PseDNC-General, PC-PseTNC-General, SC-PseDNC-General, SC-PseTNC-General, PC-PseAAC-General, SC-PseAAC-General, PC-PseAAC, SC-PseAAC methods (default=2). And For MAC, PDT, PDT-Profile, GAC, NMBAC methods (default=1). If there are several methods, enter the values in turn.
-w	The weight factor used to adjust the effect of the correlation factors. For PseSSC, PseDNC, PseKNC, PC-PseDNC-General, PC-PseTNC-General, SC-PseDNC-General, SC-PseTNC-General, PC-PseAAC-General, SC-PseAAC-General, PC-PseAAC, SC-PseAAC methods (default=0.1). If there are several methods, enter the values in turn.
-i	The index file user chosen. If there are several methods, enter the values in turn.
-e	The user-defined index file. If there are several methods, enter the values in turn.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).For Top-n-gram, PDT-Profile, DT, AC-PSSM, CC-PSSM, ACC-PSSM, PDT methods.
-lag	The value of lag. For DAC, DCC, DACC, TAC, TCC, TACC, AC, CC, ACC, ACC-PSSM, AC-PSSM and CC-PSSM methods. The value of lag (default=2). If there are several methods, enter the values in turn.
-n	The maximum distance between structure statuses, (default=0). It works with PseDPC method. Or for Top-n-gram, PDT-Profile methods. The value of top-n-gram(default=2). If there are several methods, enter the values in turn.

-f {tab, svm, csv}	The output format (default = tab). tab -- Simple format, delimited by TAB. svm -- The LIBSVM training data format. csv -- The format that can be loaded into a spreadsheet program.
-labels	The libSVM output file label. If the argument “-f” is set as “svm”, this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-ps	The input positive source file in FASTA format for IDKmer. Only for IDKmer method.
-ns	The input negative source file in FASTA format for IDKmer. Only for IDKmer method.
-max_dis	The max distance value of DR, DT, Distance Pair. Only for DR, DT and Distance Pair methods(default = 3). If there are several methods, enter the values in turn.
-cp	The reduced alphabet scheme. Choose one of the four: cp_13, cp_14, cp_19, cp_20. Only for Distance Pair method.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.

“train.py” usage

Command line arguments for “train.py”:

required	description
files	The input files. If the algorithm is set as SVM, the format of files should be LIBSVM format; if the algorithm is set as rf, the format of files should be csv format; if the algorithm is set as oet_knn or cda, the format of files should be tab format. For binary classification, two files needed. For multiclass classification, at least three files needed.
-m M	The name of the trained SVM model. Only for svm and rf.

Optional	description
-h, --help	Show this help message and exit.
-p {ACC,MCC,AUC}	The performance metric used for parameter selection. Default value is “ACC”.
-v V	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character “j”) jackknife cross validation.

-ind	The independent test dataset, The input files in FASTA format.
-ml {svm, rf, oet_knn, cda}	The method of machine learning. svm is support vector machine; rf is random forest; oet_knn is Optimized Evidence-Theoretic KNN algorithm; cda is covariance discriminant algorithm. (default is svm)
-opt	If the algorithm is set as svm: 0: small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. If the algorithm is set as rf: 0: small range set number of trees from 100 to 600, step is 200. 1: large range set number of trees from 100 to 600, step is 100. If the algorithm is set as oet_knn: 0: small range set neighbors from 1 to 30, step is 2. 1: large range set neighbors from 1 to 30, step is 1. Default value is 0.
-b {0,1}	Whether to train a SVC or SVR model for probability estimates, 0 or 1. Default value is 0.
-cpu CPU	The maximum number of CPU cores used for multiprocessing during parameter selection process. Default value is 1.
-bp {1, 0}	The option of batch processing. 1 is run batch processing, 0 is not. Default is 0.

“predict.py” usage

Command line arguments for “predict.py”:

required	description
inputfiles	The input files in LIBSVM format.
-m M	The name of the trained SVM model.

optional	description
-h, --help	Show this help message and exit.
-labels LABELS	The real label file. Optional.
-ml {svm, rf }	The method of machine learning. rf is Random Forest. (default is svm)

-o O	The output file name listing the predicted labels. The default name is “output_labels.txt”.

“ensemble.py” usage

Command line arguments for “ensemble.py”:

required	description
inputfile	The input file in tab format.
-labels LABELS	The real label file.
-classif	The module files trained in train.py or analysis.py.

optional	description
-h, --help	Show this help message and exit.
-labels LABELS	The real label file. Optional.
-w	The weights of the classifiers. Default values are all 1.0.

“analysis.py” usage

Command line arguments for “analysiss.py”:

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.
{DNA, RNA, Protein}	The sequence type.
-model	The name of the trained model.
-method	The method names. You can input several methods. The vector of each method implements linear merging. Up to 3 methods.

Optional	description
-h, --help	Show this help message and exit.
-b{0, 1}	Whether to train a SVC or SVR model for probability estimates, 0 or 1.(default=0). For svm method.
-v	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character “j”) jackknife cross validation.
-opt	Set the range of parameters to be optimized. 0: For svm, small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. For random forest, trees from 100 to 600, step is 200. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. For random forest, trees from 100 to 600, step is 100. (default=0).
-p {ACC,MCC,AUC}	The performance metric used for parameter selection. Default value is “ACC”.
-ind	The independent test dataset, The input files in FASTA format.

-out	The output files used for storing results. The number of output files should be the same as that of input files.
-k K	The number of k adjacent structure statuses. (For PseKNC and Mismatch, default is from 1 to 4. For Kmer, RevKmer, IDKmer, PseSSC and Subsequence, default is from 1 to 3.). If there are several methods, enter the ranges in turn.
-m M	For Mismatch. The max value inexact matching. (m<k) (default is from 1 to 4). If there are several methods, enter the ranges in turn.
-delta	For subsequence method. The value of penalized factor. (0<=delta<=1) (default is from 0 to 0.8). If there are several methods, enter the ranges in turn.
-a {True, False}	Choose or do not choose all physicochemical indices, default=False.
-r	Whether consider the reverse complement or not. 1 means True, 0 means False. For Kmer method. (default=0). Or the value of lambda, represents the highest counted rank (or tier) of the structural correlation along a RNA chain. For PseSSC, PseDPC methods. (default is from 1 to 7). If there are several methods, enter the ranges in turn.
-oli	Choose one kind of Oligonucleotide: 0 represents dinucleotide, default; 1 represents trinucleotide. For DAC, DCC, DACC, TAC, TCC, TACC, MAC, GAC, NMBAC, AC, CC, ACC methods.
-lamada	The value of lamada. For PseDNC, PseKNC, PC-PseDNC-General, PC-PseTNC-General, SC-PseDNC-General, SC-PseTNC-General, PC-PseAAC-General, SC-PseAAC-General, PC-PseAAC, SC-PseAAC, MAC, PDT, PDT-Profile, GAC, NMBAC methods (default is from 1 to 7). If there are several methods, enter the ranges in turn.
-w	The weight factor used to adjust the effect of the correlation factors. For PseSSC, PseDNC, PseKNC, PC-PseDNC-General, PC-PseTNC-General, SC-PseDNC-General, SC-PseTNC-General, PC-PseAAC-General, SC-PseAAC-General, PC-PseAAC, SC-PseAAC methods (default is from 0.1 to 0.8). If there are several methods, enter the
-i	The index file user chosen.
-e	The user-defined index file.

-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).For Top-n-gram, PDT-Profile, DT, AC-PSSM, CC-PSSM, ACC-PSSM, PDT methods and the number of CPU cores used for multiprocessing during parameter selection process.
-lag	The value of lag. For DAC, DCC, DACC, TAC, TCC, TACC, AC, CC, ACC, ACC-PSSM, AC-PSSM and CC-PSSM methods. The value of lag (default is from 1 to 7). If there are several methods, enter the ranges in turn.
-n	The maximum distance between structure statuses, (default is from 1 to 4). It works with PseDPC method. Or for Top-n-gram, PDT-Profile methods. The value of top-n-gram (default is from 1 to 2). If there are several methods, enter the ranges in turn.
-ml {svm, rf, oet_knn, cda}	The method of machine learning. rf is Random Forest. Oet_knn is Optimized Evidence-Theoretic K-Nearest Neighbor. Cda is covariance discriminant algorithm (default is svm)
-rl	The real label file. Optional.
-labels	The libSVM output file label. If the argument "-f " is set as "svm", this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'; For multiclass classification problem, the labels can be set as integers.
-ps	The input positive source file in FASTA format for IDKmer. Only for IDKmer method.
-ns	The input negative source file in FASTA format for IDKmer. Only for IDKmer method.
-max_dis	The max distance value of DR, DT, Distance Pair. Only for DR, DT and Distance Pair methods(default is from 1 to 4). If there are several methods, enter the ranges in turn.
-cp	The reduced alphabet scheme. Choose one of the four: cp_13, cp_14, cp_19, cp_20. Only for Distance Pair method.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.

"optimization.py" usage

Command line arguments for "optimization.py":

Required	description
inputfiles	The input files in FASTA format. More than one file could be input.

{DNA, RNA, Protein}	The sequence type.
-model	The name of the trained model.

Optional	description
-h, --help	Show this help message and exit.
-v	The cross validation mode. n: (an integer larger than 0) n-fold cross validation. j: (character "j") jackknife cross validation.
-opt	Set the range of parameters to be optimized. 0: For svm, small range set c from -5 to 10, step is 2; g from -10 to 5, step is 2. For random forest, trees from 100 to 600, step is 200. 1: large range set c from -5 to 10, step is 1; g from -10 to 5, step is 1. For random forest, trees from 100 to 600, step is 100. (default=0).
-out	The output files used for storing results. The number of output files should be the same as that of input files.
-cpu	The maximum number of CPU cores used for multiprocessing in generating frequency profile. (default=1).For Top-n-gram, PDT-Profile, DT, AC-PSSM, CC-PSSM, ACC-PSSM, PDT methods and the number of CPU cores used for multiprocessing during parameter selection process.
-ml { svm, rf, oet_knn, _cda }	The method of machine learning. rf is Random Forest. Oet_knn is Optimized Evidence-Theoretic K-Nearest Neighbor. Cda is covariance discriminant algorithm (default is svm)
-labels	The libSVM output file label. If the argument "-f " is set as "svm", this argument is required. And the number of labels should be the same as that of the input files. For binary classification problem, the labels should be '+1' or '-1'.
-sp {over, under, none}	Balance the unbalanced data, default value is none. Over is oversampling technique. Under is under sampling technique.
-bp {1, 0}	The option of batch processing. 1 is batch processing, 0 is not. Default is 0.

Example

Four examples of using BioSeq-Analysis-Seq to construct machine learning predictor for solving a specific task in bioinformatics are given.

Example of DNA

Reconstructing the predictor iDHS-EL for identification DNase I hypersensitive sites by fusing three different modes of pseudo nucleotide composition based on the benchmark dataset (22) by using BioSeq-Analysis-Seq.

The benchmark dataset contains 280 positive samples and 737 negative samples. The benchmark dataset are available at here

In this example, the files "dna_pos.txt" and "dna_neg.txt" contain the positive dataset and negative dataset of the benchmark dataset, respectively. All these two files are available in the “/data/example” folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/dna_pos.txt ./data/example/dna_neg.txt DNA
-method Kmer Kmer PseDNC -ml rf -k 1 3 1 3 -lamada 1 3 -w 0.1 0.2 -r 0 1 -labels
+1 -1 -model dna.model -opt 0 -v 5 -cpu 2
```

The output informations is as follows:

```
Processing...
MMethod Kmer is calculating...k is 1 trees are 100ethod Kmer is calculating...k is 1
trees are 300

The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv_Kmer_k_1.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv_Kmer_k_1.txt
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv_Kmer_k_1.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv_Kmer_k_1.txt
Method Kmer is calculating...k is 1 trees are 500
Method Kmer is calculating...k is 2 trees are 100
Method Kmer is calculating...k is 2 trees are 300
Method Kmer is calculating...k is 2 trees are 500
Method Kmer is calculating...k is 3 trees are 100
The output file(s) can be found here:
C:\Users\Downloads\
BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_csv_Kmer_k_3.txt
C:\Users\Downloads\
BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_csv_Kmer_k_3.txt
Method Kmer is calculating...k is 3 trees are 300
Method Kmer is calculating...k is 3 trees are 500

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv_Kmer_k_2.txt

The output file(s) with the best params can be found here:
……
……
……
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv_PseDNC_lamada_3_w_0.2.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv_PseDNC_lamada_3_w_0.2.txt
Method PseDNC is calculating...lamada is 3 w is 0.20 trees are 300
Method PseDNC is calculating...lamada is 3 w is 0.20 trees are 500
```

```
The output file(s) with the best params can be found here:
C:\Users\
Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_csv_PseD
NC_lamada_1_w_0.2.txt

The output file(s) with the best params can be found here:
C:\Users\Downloads\
BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_csv_PseDNC_lamada
_1_w_0.2.txt
Parameters selecting of features done!


Combine the features of given methods and train it...
Method Kmer is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv.txt
Method Kmer is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv.txt
Method PseDNC is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_pos_
csv.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\dna_neg_
csv.txt
Processing...
Parameter selection is in processing...
Trees are 100...
Trees are 300...
Trees are 500...

The time cost for parameter selection is 22.30s
Parameter selection of Random Forest completed.

The optimal parameters for the dataset is: Trees = 500


Model training is in processing...
The cross validation results are as follows:
ACC = 0.8514
MCC = 0.6084
AUC = 0.8311
Sn   = 0.6607
Sp   = 0.9239

The ROC curve has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\cv_ro
c.png
```

```
Model training completed.
The model has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\dna.model

Total used time: 234.78s
```

The generated ROC curve is shown in **Fig. 1**.

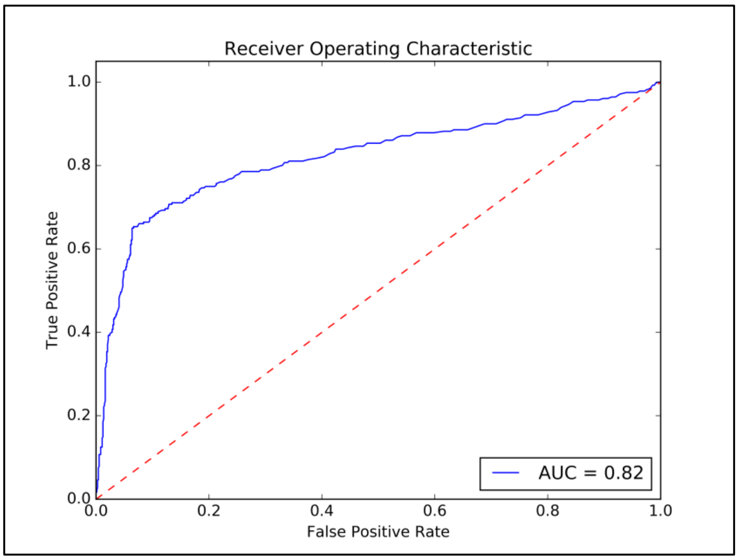


Fig .1. The ROC curve of cross validation

As shown in this example, the iDHS-EL can be easily constructed based on the benchmark dataset by using the script "analysis.py".

Example of RNA

Reconstructing the predictor iMcRNA-PseSSC for identification of real microRNA precursors based on the benchmark dataset (22) by using **BioSeq-Analysis-Seq**.
The benchmark dataset contains 1612 positive samples and 1612 negative samples. The benchmark dataset are available at here.

In this example, the files "rna_pos_with_2rd_structure.txt" and "rna_neg_with_2rd_structure.txt" contain the positive dataset and negative dataset of the benchmark dataset, respectively. All these two files are available in the "/data/example" folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/rna_pos_with_2rd_structure.txt ./data/example/
rna_neg_with_2rd_structure.txt RNA -method PseSSC -k 1 2 -r 5 6 -w 0.4 0.6 -ml
svm -labels +1 -1 -model rna.model -opt 0 -v 5 -cpu 4
```

The output informations is as follows:

```
Processing...
Method Kmer is calculating...k is 1 c is -5 g is -10M
ethod Kmer is calculating...k is 1 c is -5 g is -7
The output file(s) can be found here:
```

```
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_pos_s
vm_Kmer_k_1.txthe output file(s) can be found here:

C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_pos_s
vm_Kmer_k_1.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_neg_s
vm_Kmer_k_1.txt:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\ex
ample\rna_neg_svm_Kmer_k_1.txt

Method Kmer is calculating...k is 1 c is -5 g is -4
Method Kmer is calculating...k is 1 c is -5 g is -1
Method Kmer is calculating...k is 1 c is -5 g is 2
Method Kmer is calculating...k is 1 c is -5 g is 5
Method Kmer is calculating...k is 1 c is -2 g is -10
Method Kmer is calculating...k is 1 c is -2 g is -7
Method Kmer is calculating...k is 1 c is -2 g is -4
Method Kmer is calculating...k is 1 c is -2 g is -1
Method Kmer is calculating...k is 1 c is -2 g is 2
……
……
……
Method Kmer is calculating...k is 1 c is 10 g is -10
Method Kmer is calculating...k is 1 c is 10 g is -7
Method Kmer is calculating...k is 1 c is 10 g is -4
Method Kmer is calculating...k is 1 c is 10 g is -1
Method Kmer is calculating...k is 1 c is 10 g is 2
Method Kmer is calculating...k is 1 c is 10 g is 5
Method Kmer is calculating...k is 2 c is -5 g is -10
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_pos_s
vm_Kmer_k_2.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_neg_s
vm_Kmer_k_2.txt
Method Kmer is calculating...k is 2 c is -5 g is -7
Method Kmer is calculating...k is 2 c is -5 g is -4
Method Kmer is calculating...k is 2 c is -5 g is -1
Method Kmer is calculating...k is 2 c is -5 g is 2
Method Kmer is calculating...k is 2 c is -5 g is 5
Method Kmer is calculating...k is 2 c is -2 g is -10
Method Kmer is calculating...k is 2 c is -2 g is -7
……
……
Method Kmer is calculating...k is 2 c is 7 g is -1
Method Kmer is calculating...k is 2 c is 7 g is 2
Method Kmer is calculating...k is 2 c is 7 g is 5
Method Kmer is calculating...k is 2 c is 10 g is -10
Method Kmer is calculating...k is 2 c is 10 g is -7
Method Kmer is calculating...k is 2 c is 10 g is -4
Method Kmer is calculating...k is 2 c is 10 g is -1
Method Kmer is calculating...k is 2 c is 10 g is 2
Method Kmer is calculating...k is 2 c is 10 g is 5

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_pos_s
```

```
vm_Kmer_k_2.txt

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_neg_s
vm_Kmer_k_2.txt
Parameters selecting of features done!


Combine the features of given methods and train it...
Method Kmer is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_pos_s
vm.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\rna_neg_s
vm.txt
Processing on the best params...
Parameter selection is in processing...

Iteration   c =   10   g =   -7   finished.
Iteration   c =   -5   g =   -1   finished.
Iteration   c =   4   g =   -1   finished.
Iteration   c =   4   g =   2   finished.
Iteration   c =   4   g =   -4   finished.
Iteration   c =   -2   g =   -4   finished.
Iteration   c =   7   g =   -7   finished.
Iteration   c =   1   g =   -4   finished.
Iteration   c =   -5   g =   -4   finished.
Iteration   c =   4   g =   5   finished.
……
……
……
Iteration   c =   -5   g =   5   finished.
Iteration   c =   1   g =   -1   finished.
Iteration   c =   -5   g =   2   finished.
Iteration   c =   1   g =   -10   finished.
Iteration   c =   1   g =   2   finished.
Iteration   c =   7   g =   5   finished.
Iteration   c =   7   g =   -4   finished.
Iteration   c =   10   g =   2   finished.
The time cost for parameter selection is 74.15s
Parameter selection completed.

The optimal parameters for the dataset are: C =   16   gamma =   4


Model training is in processing...
The cross validation results are as follows:
ACC = 0.7212
MCC = 0.4435
AUC = 0.7894
Sn   = 0.6887
Sp   = 0.7546

The ROC curve has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\cv_ro
```

```
c.png

Model training completed.
The model has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\rna.model

Done.
Used time: 80.52s
Total used time: 171.21s
```

The generated ROC curve is shown in **Fig. 2**.

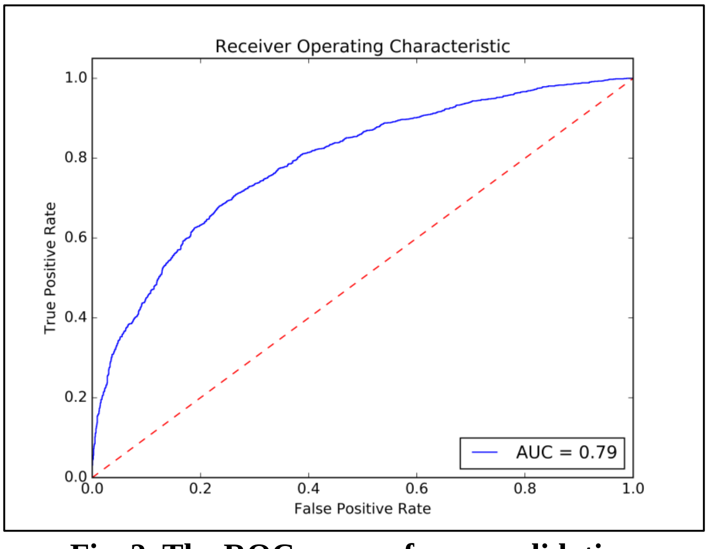


Fig .2. The ROC curve of cross validation

As shown in this example, the iMcRNA-PseSSC can be easily constructed based on the benchmark dataset by using the script "analysis.py".

Example of protein

Reconstructing the predictor PseDNA-Pro for DNA binding protein identification based on the benchmark dataset (22), and evaluating its performance on an independent dataset (29) by using **BioSeq-Analysis-Seq.**

The benchmark dataset contains 525 positive samples and 550 negative samples. There are 93 positive samples and 93 negative samples in the independent dataset. The benchmark dataset and independent dataset are available at benchmark dataset and independent dataset, respectively.

In this example, the files "protein_pos.txt" and "protein_neg.txt" contain the positive dataset and negative dataset of the benchmark dataset, respectively. The samples of the independent dataset and their labels are stored in the files "protein_test.txt" and "labels.txt", respectively. All these four files are available in the "/data/example" folder.

We can use a command to implement feature extraction and model training, while implementing optimization parameters.

```
python analysis.py ./data/example/Protein_pos.txt ./data/example/Protein_neg.txt Protein -method PC-PseAAC -lamada 2 4 -w 0.05 0.3 -ml svm -labels +1 -1 -model protein.model -opt 0 -v 5
```

The output informations is as follows:

```
Processing...
```

```
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -10
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_p
os_svm_PC-PseAAC_lamada_2_w_0.05.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_n
eg_svm_PC-PseAAC_lamada_2_w_0.05.txt
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is 5
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is 5
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is 5
……
……
……
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 4 g is 5
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is -10
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is -7
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is -4
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is -1
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is 2
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 7 g is 5
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -10
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -7
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -4
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -1
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is 2
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is 5

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_p
os_svm_PC-PseAAC_lamada_3_w_0.05.txt

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_n
eg_svm_PC-PseAAC_lamada_3_w_0.05.txt
Parameters selecting of features done!


Combine the features of given methods and train it...
Method PC-PseAAC is calculating...
The output file(s) can be found here:
```

```
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_pos_svm.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_neg_svm.txt
Processing on the best params...
Parameter selection is in processing...

Iteration   c =   7   g =   -1   finished.
Iteration   c =   4   g =   -10   finished.
Iteration   c =   4   g =   5   finished.
Iteration   c =   4   g =   -1   finished.
Iteration   c =   10   g =   -1   finished.
……
……
……
Iteration   c =   7   g =   2   finished.
Iteration   c =   -5   g =   2   finished.
Iteration   c =   4   g =   -4   finished.
Iteration   c =   -2   g =   -4   finished.
Iteration   c =   -2   g =   -1   finished.
Iteration   c =   1   g =   -1   finished.
Iteration   c =   4   g =   -7   finished.
Iteration   c =   10   g =   -4   finished.
The time cost for parameter selection is 32.54s
Parameter selection completed.

The optimal parameters for the dataset are: C =   16   gamma =   4


Model training is in processing...
The cross validation results are as follows:
ACC = 0.7526
MCC = 0.5049
AUC = 0.8177
Sn   = 0.7429
Sp   = 0.7615

The ROC curve has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\cv_roc.png

Model training completed.
The model has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\protein.model


Done.
Used time: 35.35s
Total used time: 308.27s
```

The generated ROC curve is shown in **Fig. 3**.

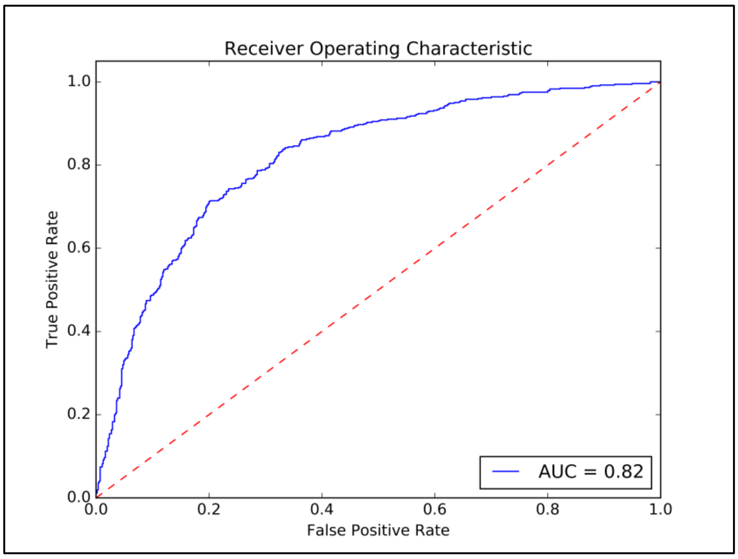


Fig .3. The ROC curve of cross validation

As shown in this example, the PseDNA-Pro can be easily constructed based on the benchmark dataset by using the script “analysis.py”.

If we want to use an independent test set to evaluate the model, we can change this command to:

```
python analysis.py ./data/example/Protein_pos.txt ./data/example/Protein_neg.txt
Protein -method PC-PseAAC -lamada 2 4 -w 0.05 0.3 -ml svm -labels +1 -1 -model
protein.model -ind ./data/example/protein_test.txt -rl ./data/example/labels.txt -opt 0
-v 5 -cpu 4
```

The output informations is as follows:

```
Processing...
MMethod PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -10ethod
PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -7

TThe output file(s) can be found here:he output file(s) can be found here:

CC:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_
pos_svm_PC-PseAAC_lamada_2_w_0.05.txt:\Users\Downloads\BioSeq-Analysis2.0\Bi
oSeq-Analysis-Seq\data\example\Protein_pos_svm_PC-PseAAC_lamada_2_w_0.05.txt

CC:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_
neg_svm_PC-PseAAC_lamada_2_w_0.05.txt:\Users\Downloads\BioSeq-Analysis2.0\Bi
oSeq-Analysis-Seq\data\example\Protein_neg_svm_PC-PseAAC_lamada_2_w_0.05.txt

Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is -1
MMethod PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is 5
ethod PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -5 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is 2M
ethod PC-PseAAC is calculating...lamada is 2 w is 0.05 c is -2 g is 5
```

```
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is -1
MMethod PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is 2ethod
PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 1 g is 5

Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 4 g is 5
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is -1
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is 2
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 7 g is 5
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 10 g is -10
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 10 g is -7
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 10 g is -4
Method PC-PseAAC is calculating...lamada is 2 w is 0.05 c is 10 g is -1
……
……
……
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -10
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -7
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -4
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is -1
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is 2
Method PC-PseAAC is calculating...lamada is 4 w is 0.35 c is 10 g is 5

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_p
os_svm_PC-PseAAC_lamada_2_w_0.35.txt

The output file(s) with the best params can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_n
eg_svm_PC-PseAAC_lamada_2_w_0.35.txt
Parameters selecting of features done!


Combine the features of given methods and train it...
Method PC-PseAAC is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_p
os_svm.txt
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\Protein_n
eg_svm.txt
Processing on the best params...
Parameter selection is in processing...

Iteration    c =    -5    g =    -7    finished.
Iteration    c =    -5    g =    2    finished.
```

```
Iteration    c =    -2    g =    -10    finished.
Iteration    c =    10    g =    2    finished.
Iteration    c =    4    g =    2    finished.
Iteration    c =    10    g =    5    finished.
Iteration    c =    -2    g =    2    finished.
Iteration    c =    -2    g =    5    finished.
……
……
Iteration    c =    4    g =    -10    finished.
Iteration    c =    7    g =    -1    finished.
Iteration    c =    4    g =    -7    finished.
Iteration    c =    10    g =    -10    finished.
Iteration    c =    7    g =    2    finished.
The time cost for parameter selection is 20.52s
Parameter selection completed.

The optimal parameters for the dataset are: C =    128    gamma =    4


Model training is in processing...
The cross validation results are as follows:
ACC = 0.7423
MCC = 0.4851
AUC = 0.8141
Sn    = 0.7367
Sp    = 0.7484

The ROC curve has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\cv_ro
c.png

Model training completed.
The model has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\protei
n.model

Done.
Used time: 23.44s

Predict on the independent dataset...

Method PC-PseAAC is calculating...
The output file(s) can be found here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\example\protein_te
st_svm.txt
The parameters of RBF kernel:
c =    128    g =    4
The performance evaluations are as follows:

ACC = 0.6828
MCC = 0.3692
AUC = 0.7237
Sn    = 0.7527
Sp    = 0.6129
```

```
The ROC curve has been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\predicted_roc.png

The predicted labels have been saved. You can check it here:
C:\Users\Downloads\BioSeq-Analysis2.0\BioSeq-Analysis-Seq\data\final_results\output_labels.txt

Done.
Used time: 1.30s
Total used time: 183.47s
```

2.5 Methods description

2.5.1 Feature extraction

The **BioSeq-Analysis-Seq** stand-alone package is able to generate totally 56 different modes of pseudo components for DNA, RNA, and protein sequences, including 20 modes for DNA sequences (**Table 1-b**), 14 modes for RNA sequences (**Table 2-b**), and 22 modes for protein sequences (**Table 3-b**). The detailed information of the 56 methods will be introduced in BioSeq-Analysis-Seq description document which can be downloaded from here: http://bliulab.net/BioSeq-Analysis2.0/doc/ .

For many biological sequence analysis tasks, the training sets are imbalanced. As a result, a predictor trained by a skewed dataset would inevitably lead to a bias consequence (24). The oversampling and undersampling are widely used to minimize this bias consequence. For undersampling, some samples are randomly removed from the large class to make the number of samples in different classes the same. For the oversampling, some hypothetical samples are inserted into the small classes in order to make each class with equal number of samples. In **BioSeq-Analysis-Seq**, the SMOTE algorithm (25) were employed to generate the hypothetical samples for this purpose.

2.5.2 Parameter selection

In LIBSVM there are two parameters *c* and *g* which can determine the performance of the predictor. In Random Forest there is one parameter *t* which can determine the performance of the predictor. In OET-KNN, there is one parameter *k* which can determine the performance of the predictor. Each method of the 56 methods achieved in stand-alone package has respective parameters, such as the Kmer method has parameter “k”. **BioSeq-Analysis-Seq** is able to automatically optimize these parameters based on the best performance on the validation set. Users can choose a range of the parameters for optimizing. For more information of the input format, please refer to “**Commands**” section.

To improve the efficiency of this procedure, multiprocessing technique is applied, which significantly reduces the computational cost. One of the three performance measures, including Accuracy (ACC), Mathew’s Correlation Coefficient (MCC) and Area Under roc Curve (AUC) can be used as the golden standard to optimize the parameters.

2.5.3 Predictor construction

In the model training process, this model is trained based on LIBSVM with RBF kernel, Random Forest, and two lazy learning algorithms: OET-KNN and Covariance Discriminant.

2.5.4 Cross validation

BioSeq-Analysis-Seq provides three types of cross validation options, including k-fold cross validation, jackknife (leave-one-out cross validation) and independent dataset test, which can be chosen by the argument “-v”. Please refer to “**Commands**” section for

more details.
For binary classification, the performance of the predictor is measured by five common performance measures, including the accuracy (ACC), Mathew's Correlation Coefficient (MCC), Area Under roc Curve (AUC), sensitivity (Sn), and specificity (Sp). Furthermore, the ROC (Receiver Operating Characteristic) (26) curve will also be generated and saved in a PNG file.
For multiclass classification, only the performance measure of ACC is calculated since the other measures are not suitable for multiclass classification.
Besides, if the parameter "-b" of libsvm is set or using the random forest, the prediction probability values will be output and save as a file, thus users can do further analysis with these data.

2.5.5 Sequence prediction

The "predict.py" is used to predict the unseen samples based on the model trained by using "train.py". The performance of the predictors can be further evaluated on the independent datasets. If the label information of the independent dataset is not available, the performance of the predictor will not be evaluated, and only the predicted labels are given. Otherwise, this script will output the predicted labels. For binary classification, the five performance measures (ACC, MCC, AUC, Sn, and Sp) will be calculated along with the corresponding ROC curve saved as a PNG file; for multiclass classification, only the performance measure ACC will be calculated.

2.5.6 Ensemble learning

Sometimes one predictor may not achieve the expected results. By combining several different predictors, better prediction performance could be obtained. Thus, ensemble learning has been widely used. The stand-alone package of **BioSeq-Analysis-Seq** provides a script "ensemble.py" used for ensemble learning based on the predictors generated by "train.py" or "analysis.py".

Table 1-a. 7 residue-level modes for DNA sequences.

Category	Mode	Description
Residue composition	One-hot	Basic one-hot (30)
	Position-specific-2	Position-specific of two nucleotides (31)
	Position-specific-3	Position-specific of three nucleotides (31)
	Position-specific-4	Position-specific of four nucleotides(31)
Physicochemical property	DPC	Dinucleotide physicochemical (32,33)
	TPC	Trinucleotide physicochemical (32,33)
Evolutionary information	BLAST-matrix	BLAST-matrix (34)

Table 1-b. 20 sequence-level modes for DNA sequences.

Category	Mode	Description
Nucleic acid Composition	Kmer	Basic kmer (35)
	RevKmer	Reverse complementary kmer(36,37)
	IDKmer	increment of diversity (38-40)
	Mismatch	The occurrences of kmers, allowing at most m mismatches (41-43)
	Subsequence	The occurrences of kmers, allowing non-contiguous matches (41,43,44)
Autocorrelation	DAC	Dinucleotide-based auto covariance (45,46)
	DCC	Dinucleotide-based cross covariance (45,46)
	DACC	Dinucleotide-based auto-cross covariance (45,46)
	TAC	Trinucleotide-based auto covariance (45)
	TCC	Trinucleotide-based cross covariance (45)
	TACC	Trinucleotide-based auto-cross covariance (45)
	MAC	Moran autocorrelation (47,48)
	GAC	Geary autocorrelation (48,49)
	NMBAC	Normalized Moreau-Broto autocorrelation (48,50)
Pseudo nucleotide composition	PseDNC	Pseudo dinucleotide composition (51)
	PseKNC	Pseudo k-tuple nucleotide composition (52,53)
	PC-PseDNC-General	General parallel correlation pseudo dinucleotide composition (54)
	PC-PseTNC-General	General parallel correlation pseudo trinucleotide composition (54)
	SC-PseDNC-General	General series correlation pseudo dinucleotide composition (54)
	SC-PseTNC-General	General series correlation pseudo trinucleotide composition (54)

Table 2-a. 6 residue-level modes for RNA sequences.

Category	Mode	Description
Residue composition	One-hot	Basic one-hot (30)
	Position-specific-2	Position-specific of two nucleotides (31)

	Position-specific-3	Position-specific of three nucleotides (31)
	Position-specific-4	Position-specific of four nucleotides(31)
Physicochemical property	DPC	Dinucleotide physicochemical (32,33)
Structure composition	SS	Secondary structure (55)

Table 2-b. 14 sequence-level modes for RNA sequences.

Category	Mode	Description
Nucleic acid Composition	Kmer	Basic kmer (53)
	Mismatch	The occurrences of kmers, allowing at most m mismatches (41-43)
	Subsequence	The occurrences of kmers, allowing non-contiguous matches (41,43,44)
Autocorrelation	DAC	Dinucleotide-based auto covariance (45,46,56)
	DCC	Dinucleotide-based cross covariance (45,46,56)
	DACC	Dinucleotide-based auto-cross covariance (45,46,56)
	MAC	Moran autocorrelation (47,48)
	GAC	Geary autocorrelation (48,49)
	NMBAC	Normalized Moreau-Broto autocorrelation (48,50)
Pseudo nucleotide composition	PC-PseDNC- General	General parallel correlation pseudo dinucleotide composition (46,48)
	SC-PseDNC-General	General series correlation pseudo dinucleotide composition (46,48)
Predicted Structure composition	Triplet	Local structure-sequence triplet element (57)
	PseSSC	Pseudo-structure status composition (22)
	PseDPC	Pseudo-distance structure status pair composition (58)

Table 3-a. 13 residue-level modes for protein sequences

Category	Mode	Description
Residue composition	One-hot	Basic one-hot (30)

	One-hot(6-bit)	6-dimension One-hot method (59)
	Binary(5-bit)	Use five binary bit to encode (60)
	AESNN3	Learn from alignments (61)
	Position-specific-2	Position-specific of two residues (31)
Physicochemical property	PP	Properties form AAindex (62)
Structure composition	SS	Secondary structure (63)
	SASA	Solvent accessible surface area (64)
Evolutionary information	PAM250	PAM250 matrix (65)
	BLOSUM62	BLOSUM62 matrix (66)
	PSSM	PSSM matrix (67)
	PSFM	Frequency profiles matrix (68)
	CS	Conservation score (69)

Table 3-b. 22 sequence-level modes for protein sequences.

Category	Mode	Description
Amino acid composition	Kmer	Basic kmer (70)
	DR	Distance-based Residue (71)
	Distance Pair	PseAAC of Distance-Pairs and Reduced Alphabet (72)
Autocorrelation	AC	Auto covariance (45,56)
	CC	Cross covariance (45,56)
	ACC	Auto-cross covariance (45,56)
	PDT	Physicochemical distance transformation (73)
Pseudo amino acid composition	PC-PseAAC	Parallel correlation pseudo amino acid composition (74)
	SC-PseAAC	Series correlation pseudo amino acid composition (75)
	PC-PseAAC-General	General parallel correlation pseudo amino acid composition (74,76)
	SC-PseAAC-General	General series correlation pseudo amino acid composition (75,76)
Profile-based features	Top-n-gram	Select and combine the n most frequent amino acids

		according to their frequencies. (70)
	PDT-Pofile	Profile-based Physicochemical distance transformation (73)
	DT	Distance-based Top-n-gram (71)
	AC-PSSM	Profile-based Auto covariance (45)
	CC-PSSM	Profile-based Cross covariance (45)
	ACC-PSSM	Profile-based Auto-cross covariance (45)
	PSSM-DT	PSSM distance transformation (77)
	PSSM-RT	PSSM relation transformation (78)
	CS	sequence conservation score (69)
Predicted structure features	SS	secondary structure (63)
	SASA	solvent accessible surface area (64)

Table 4. The names of the 148 physicochemical indices for dinucleotides.

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 5. The names of the 12 physicochemical indices for trinucleotides.

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

Table 6. The names of the 90 physicochemical indices for dinucleotides.

Base stacking	Protein induced deformability	B-DNA twist
Dinucleotide GC Content	A-philicity	Propeller twist
Duplex stability-free energy	Duplex stability-disrupt energy	DNA denaturation
Bending stiffness	Protein DNA twist	Stabilising energy of Z-DNA
Aida_BA_transition	Breslauer_dG	Breslauer_dH
Breslauer_dS	Electron_interaction	Hartman_trans_free_energy
Helix-Coil_transition	Ivanov_BA_transition	Lisser_BZ_transition
Polar_interaction	SantaLucia_dG	SantaLucia_dH
SantaLucia_dS	Sarai_flexibility	Stability
Stacking_energy	Sugimoto_dG	Sugimoto_dH

Sugimoto_dS	Watson-Crick_interaction	Twist
Tilt	Roll	Shift
Slide	Rise	Stacking energy
Bend	Tip	Inclination
Major Groove Width	Major Groove Depth	Major Groove Size
Major Groove Distance	Minor Groove Width	Minor Groove Depth
Minor Groove Size	Minor Groove Distance	Persistance Length
Melting Temperature	Mobility to bend towards major groove	Mobility to bend towards minor groove
Propeller Twist	Clash Strength	Enthalpy
Free energy	Twist_twist	Tilt_tilt
Roll_roll	Twist_tilt	Twist_roll
Tilt_roll	Shift_shift	Slide_slide
Rise_rise	Shift_slide	Shift_rise
Slide_rise	Twist_shift	Twist_slide
Twist_rise	Tilt_shift	Tilt_slide
Tilt_rise	Roll_shift	Roll_slide
Roll_rise	Slide stiffness	Shift stiffness
Roll stiffness	Rise stiffness	Tilt stiffness
Twist stiffness	Wedge	Direction
Flexibility_slide	Flexibility_shift	Entropy

Table 7. The names of the 6 physicochemical indices for dinucleotides.

Twist	Tilt	Roll
Shift	Slide	Rise

Table 8. The names of the 22 physicochemical indices for dinucleotides.

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)

Table 9. The names of the 11 physicochemical indices for dinucleotides.

Shift	Slide	Rise
Tilt	Roll	Twist
Stacking energy	Enthalpy	Entropy
Free energy	Hydrophilicity	

Table 10. The names of the 547 physicochemical indices for amino acids.

Hydrophobicity	Hydrophilicity	Mass
ARGP820102	ARGP820103	BEGF750101
BHAR880101	BIGC670101	BIOV880101
BROC820102	BULH740101	BULH740102
BUNA790103	BURA740101	BURA740102
CHAM820102	CHAM830101	CHAM830102
CHAM830105	CHAM830106	CHAM830107

CHOC760101	CHOC760102	CHOC760103
CHOP780201	CHOP780202	CHOP780203
CHOP780206	CHOP780207	CHOP780208
CHOP780211	CHOP780212	CHOP780213
CHOP780216	CIDH920101	CIDH920102
CIDH920105	COHE430101	CRAJ730101
DAWD720101	DAYM780101	DAYM780201
EISD840101	EISD860101	EISD860102
FASG760102	FASG760103	FASG760104
FAUJ880101	FAUJ880102	FAUJ880103
FAUJ880106	FAUJ880107	FAUJ880108
FAUJ880111	FAUJ880112	FAUJ880113
FINA910102	FINA910103	FINA910104
GEIM800102	GEIM800103	GEIM800104
GEIM800107	GEIM800108	GEIM800109
GOLD730101	GOLD730102	GRAR740101
GUYH850101	HOPA770101	HOPT810101
HUTJ700103	ISOY800101	ISOY800102
ISOY800105	ISOY800106	ISOY800107
JANJ780102	JANJ780103	JANJ790101
JOND750102	JOND920101	JOND920102
KANM800101	KANM800102	KANM800103
KARP850102	KARP850103	KHAG800101
KRIW790101	KRIW790102	KRIW790103
LEVM760101	LEVM760102	LEVM760103
LEVM760106	LEVM760107	LEVM780101
LEVM780104	LEVM780105	LEVM780106
LIFS790102	LIFS790103	MANP780101
MAXF760103	MAXF760104	MAXF760105
MEEJ800101	MEEJ800102	MEEJ810101
MEIH800102	MEIH800103	MIYS850101
NAGK730103	NAKH900101	NAKH900102
NAKH900105	NAKH900106	NAKH900107
NAKH900110	NAKH900111	NAKH900112
NAKH920102	NAKH920103	NAKH920104
NAKH920107	NAKH920108	NISK800101
OOBM770101	OOBM770102	OOBM770103
OOBM850101	OOBM850102	OOBM850103
PALJ810101	PALJ810102	PALJ810103
PALJ810106	PALJ810107	PALJ810108
PALJ810111	PALJ810112	PALJ810113
PALJ810116	PARJ860101	PLIV810101
PONP800103	PONP800104	PONP800105
PONP800108	PRAM820101	PRAM820102
PRAM900102	PRAM900103	PRAM900104
QIAN880101	QIAN880102	QIAN880103
QIAN880106	QIAN880107	QIAN880108
QIAN880111	QIAN880112	QIAN880113
QIAN880116	QIAN880117	QIAN880118
QIAN880121	QIAN880122	QIAN880123
QIAN880126	QIAN880127	QIAN880128
QIAN880131	QIAN880132	QIAN880133
QIAN880136	QIAN880137	QIAN880138

RACS770102	RACS770103	RACS820101
RACS820104	RACS820105	RACS820106
RACS820109	RACS820110	RACS820111
RACS820114	RADA880101	RADA880102
RADA880105	RADA880106	RADA880107
RICJ880102	RICJ880103	RICJ880104
RICJ880107	RICJ880108	RICJ880109
RICJ880112	RICJ880113	RICJ880114
RICJ880117	ROBB760101	ROBB760102
ROBB760105	ROBB760106	ROBB760107
ROBB760110	ROBB760111	ROBB760112
ROSG850101	ROSG850102	ROSM880101
SIMZ760101	SNEP660101	SNEP660102
SUEM840101	SUEM840102	SWER830101
TANS770103	TANS770104	TANS770105
TANS770108	TANS770109	TANS770110
VASM830103	VELV850101	VENT840101
WEBA780101	WERD780101	WERD780102
WOEC730101	WOLR810101	WOLS870101
YUTK870101	YUTK870102	YUTK870103
ZIMJ680101	ZIMJ680102	ZIMJ680103
AURR980101	AURR980102	AURR980103
AURR980106	AURR980107	AURR980108
AURR980111	AURR980112	AURR980113
AURR980116	AURR980117	AURR980118
ONEK900101	ONEK900102	VINM940101
VINM940104	MUNV940101	MUNV940102
MUNV940105	WIMW960101	KIMC930101
PARS000101	PARS000102	KUMS000101
KUMS000104	TAKK010101	FODM020101
NADH010103	NADH010104	NADH010105
MONM990201	KOEP990101	KOEP990102
CEDJ970103	CEDJ970104	CEDJ970105
FUKS010103	FUKS010104	FUKS010105
FUKS010108	FUKS010109	FUKS010110
AVBF000101	AVBF000102	AVBF000103
AVBF000106	AVBF000107	AVBF000108
MITS020101	TSAJ990101	TSAJ990102
WILM950101	WILM950102	WILM950103
GUOD860101	JURD980101	BASU050101
SUYM030101	PUNT030101	PUNT030102
GEOR030103	GEOR030104	GEOR030105
GEOR030108	GEOR030109	ZHOH040101
BAEK050101	HARY940101	PONJ960101
OLSK800101	KIDA850101	GUYH850102
GUYH850105	ROSM880104	ROSM880105
BLAS910101	CASG920101	CORJ870101
CORJ870104	CORJ870105	CORJ870106
MIYS990101	MIYS990102	MIYS990103
ENGD860101	FASG890101	TANS770101
ANDN920101	ARGP820101	TANS770106
BEGF750102	BEGF750103	VASM830101
BIOV880102	BROC820101	VHEG790101

BUNA790101	BUNA790102	WERD780103
CHAM810101	CHAM820101	WOLS870102
CHAM830103	CHAM830104	YUTK870104
CHAM830108	CHOC750101	ZIMJ680104
CHOC760104	CHOP780101	AURR980104
CHOP780204	CHOP780205	AURR980109
CHOP780209	CHOP780210	AURR980114
CHOP780214	CHOP780215	AURR980119
CIDH920103	CIDH920104	VINM940102
CRAJ730102	CRAJ730103	MUNV940103
DESM900101	DESM900102	MONM990101
EISD860103	FASG760101	KUMS000102
FASG760105	FAUJ830101	NADH010101
FAUJ880104	FAUJ880105	NADH010106
FAUJ880109	FAUJ880110	CEDJ970101
FINA770101	FINA910101	FUKS010101
GARJ730101	GEIM800101	FUKS010106
GEIM800105	GEIM800106	FUKS010111
GEIM800110	GEIM800111	AVBF000104
GRAR740102	GRAR740103	AVBF000109
HUTJ700101	HUTJ700102	COSI940101
ISOY800103	ISOY800104	WILM950104
ISOY800108	JANJ780101	BASU050102
JANJ790102	JOND750101	GEOR030101
JUKT750101	JUNJ780101	GEOR030106
KANM800104	KARP850101	ZHOH040102
KLEP840101	KRIW710101	DIGM050101
KYTJ820101	LAWE840101	GUYH850103
LEVM760104	LEVM760105	JACR890101
LEVM780102	LEVM780103	CORJ870102
LEWP710101	LIFS790101	CORJ870107
MAXF760101	MAXF760102	MIYS990104
MAXF760106	MCMT640101	TANS770102
MEEJ810102	MEIH800101	TANS770107
NAGK730101	NAGK730102	VASM830102
NAKH900103	NAKH900104	WARP780101
NAKH900108	NAKH900109	WERD780104
NAKH900113	NAKH920101	WOLS870103
NAKH920105	NAKH920106	ZASB820101
NISK860101	NOZY710101	ZIMJ680105
OOBM770104	OOBM770105	AURR980105
OOBM850104	OOBM850105	AURR980110
PALJ810104	PALJ810105	AURR980115
PALJ810109	PALJ810110	AURR980120
PALJ810114	PALJ810115	VINM940103
PONP800101	PONP800102	MUNV940104
PONP800106	PONP800107	BLAM930101
PRAM820103	PRAM900101	KUMS000103
PTIO830101	PTIO830102	NADH010102
QIAN880104	QIAN880105	NADH010107
QIAN880109	QIAN880110	CEDJ970102
QIAN880114	QIAN880115	FUKS010102
QIAN880119	QIAN880120	FUKS010107

QIAN880124	QIAN880125	FUKS010112
QIAN880129	QIAN880130	AVBF000105
QIAN880134	QIAN880135	YANJ020101
QIAN880139	RACS770101	PONP930101
RACS820102	RACS820103	KUHL950101
RACS820107	RACS820108	BASU050103
RACS820112	RACS820113	GEOR030102
RADA880103	RADA880104	GEOR030107
RADA880108	RICJ880101	ZHOH040103
RICJ880105	RICJ880106	WOLR790101
RICJ880110	RICJ880111	GUYH850104
RICJ880115	RICJ880116	COWR900101
ROBB760103	ROBB760104	CORJ870103
ROBB760108	ROBB760109	CORJ870108
ROBB760113	ROBB790101	MIYS990105
ROSM880102	ROSM880103	SNEP660104
SNEP660103		

Table 11. The names of the 3 physicochemical indices for amino acids.

Hydrophobicity	hydrophilicity	mass

Table 12. The names of the 2 physicochemical indices for amino acids.

Hydrophobicity	hydrophilicity

References

1. Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine learning*, **20**, 273-297.
2. Ho, T.K. (1995), *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. IEEE, Vol. 1, pp. 278-282.
3. Ho, T.K. (1998) The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, **20**, 832-844.
4. Dong, Z.J., Wang, K.Y., Dang, T.K.L., Gultas, M., Welter, M., Wierschin, T., Stanke, M. and Waack, S. (2014) CRF-based models of protein surfaces improve protein-protein interaction site predictions. *Bmc Bioinformatics*, **15**.
5. Chang, C.C. and Lin, C.J. (2011) LIBSVM: A Library for Support Vector Machines. *Acm T Intel Syst Tec*, **2**, 1-27.
6. FlexCRFs: Flexible Conditional Random Fields. *Available online: http://flexcrfs.sourceforge.net/documents.html*.
7. Williams, T. and Kelley, C. (2006) Gnuplot: an interactive plotting program. *Mourrain Ufk*.
8. Van Der Walt, S., Colbert, S.C. and Varoquaux, G. (2011) The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, **13**, 22-30.
9. Jones, E., Oliphant, T. and Peterson, P. (2014) {SciPy}: open source scientific tools for {Python}.
10. Hunter, J.D. (2007) Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, **9**, 90-95.
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. and Dubourg, V. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830.
12. Lemaitre, G., Nogueira, F. and Aridas, C.K. (2017) Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning.
13. Mckinney, W. (2011) pandas: a Foundational Python Library for Data Analysis and Statistics. *Dlr De*.
14. Jones, D.T. (1999) Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, **292**, 195-202.
15. Cuff, J.A. and Barton, G.J. (2000) Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *Proteins-structure Function & Bioinformatics*, **40**, 502-511.
16. Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, A., Sharma, A., Wang, J., Sattar, A., Yang, Y. and

Zhou, Y. (2015) Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Scientific Reports*, **5**, 11476.
17. Yang, Y., Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, A., Sharma, A., Wang, J., Sattar, A. and Zhou, Y. (2017) SPIDER2: A Package to Predict Secondary Structure, Accessible Surface Area, and Main-Chain Torsional Angles by Deep Neural Networks.
18. Pupko, T., Bell, R.E., Mayrose, I., Glaser, F. and Ben-Tal, N. (2002) Rate4Site: an algorithmic tool for the identification of functional regions in proteins by surface mapping of evolutionary determinants within their homologues. *Bioinformatics*, **18 Suppl 1**, S71.
19. Glaser, F., Rosenberg, Y.A., Pupko, T. and Ben, T.N. (2005) The ConSurf-HSSP database: the mapping of evolutionary conservation among homologs onto PDB structures. *Proteins-structure Function & Bioinformatics*, **58**, 610.
20. Liu, B., Li, K., Huang, D.S. and Chou, K.C. (2018) iEnhancer-EL: identifying enhancers and their strength with ensemble learning approach. *Bioinformatics*, **34**, 3835-3842.
21. Zou, Q., Sr., Xing, P., Wei, L. and Liu, B. (2018) Gene2vec: Gene Subsequence Embedding for Prediction of Mammalian N6-Methyladenosine Sites from mRNA. *RNA*.
22. Liu, B., Fang, L., Liu, F., Wang, X., Chen, J. and Chou, K.-C. (2015) Identification of real microRNA precursors with a pseudo structure status composition approach. *PloS one*, **10**, e0121501.
23. Liu, Y., Wang, X. and Liu, B. (2018) IDP(-)CRF: Intrinsically Disordered Protein/Region Identification Based on Conditional Random Fields. *Int J Mol Sci*, **19**.
24. Liu, B., Fang, Y., Huang, D.-s. and Chou, K.-C. (2018) iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformaitcs*, **34**, 33-40.
25. Lemaitre, G., Nogueira, F. and Aridas, C.K. (2017) Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, **18**, 1-5.
26. Fawcett, T. (2006) An introduction to ROC analysis. *Pattern recognition letters*, **27**, 861-874.
27. Chou, K.C. and Shen, H.B. (2006) Predicting eukaryotic protein subcellular location by fusing optimized evidence-theoretic K-nearest neighbor classifiers. *J. Proteome Res*, **5**, 1888–1897.
28. Jia, J., Zhang, L., Liu, Z., Xiao, X. and Chou, K. (2016) pSumo-CD: predicting sumoylation sites in proteins with covariance discriminant algorithm by incorporating sequence-coupled effects into general PseAAC. *Bioinformaitcs*, **32**, 3133-3141.
29. Lou, W., Wang, X., Chen, F., Chen, Y., Jiang, B. and Zhang, H. (2014) Sequence based prediction of DNA-binding proteins based on hybrid feature selection using random forest and Gaussian naive Bayes. *PLoS One*, **9**, e86703.
30. Yoo, P.D., Zhou, B.B. and Zomaya, A.Y. (2008) Machine learning techniques for protein secondary structure prediction: An overview and evaluation. *Curr Bioinform*, **3**, 74-86.
31. 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-+.
32. Friedel, M., Nikolajewa, S., Suhnel, J. and Wilhelm, T. (2009) DiProDB: a database for dinucleotide properties. *Nucleic Acids Res*, **37**, D37-D40.
33. 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.
34. Altschul, S., Madden, T., Schaffer, A., Zhang, J.H., Zhang, Z., Miller, W. and Lipman, D. (1998) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Faseb J*, **12**, A1326-A1326.
35. 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 Research*, **43**, W65-W71.
36. 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 computational biology*, **4**, e1000134.
37. 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.
38. Chen, W., Luo, L. and Zhang, L. (2010) The organization of nucleosomes around splice sites. *Nucleic acids research*, **38**, 2788-2798.
39. Chen, H., Ouseph, M.M., Li, J., Pecot, T., Chokshi, V., Kent, L., Bae, S., Byrne, M., Duran, C., Comstock, G. *et al.* (2012) Canonical and atypical E2Fs regulate the mammalian endocycle. *Nature Cell Biology*, **14**, 1192-1202.
40. Liu, B., Fang, L.Y., Liu, F.L., Wang, X.L., Chen, J.J. and Chou, K.C. (2015) Identification of Real MicroRNA Precursors with a Pseudo Structure Status Composition Approach. *Plos One*, **10**.
41. El-Manzalawy, Y., Dobbs, D. and Honavar, V. (2008) Predicting flexible length linear B-cell epitopes. *Computational Systems Bioinformatics*, **7**, 121-132.
42. 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.
43. 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.
44. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C. (2002) Text classification using string kernels. *Journal of Machine Learning Research*, **2**, 419-444.
45. 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.
46. Friedel, M., Nikolajewa, S., Sühnel, J. and Wilhelm, T. (2009) DiProDB: a database for dinucleotide properties. *Nucleic acids research*, **37**, D37-D40.
47. Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, **27**, 451-477.
48. Chen, W., Zhang, X., Brooker, J., Lin, H., Zhang, L. and Chou, K.-C. (2015b) PseKNC-General: a cross-platform package for generating various modes of pseudo nucleotide compositions. *Bioinformatics*, **31**, 119-120.
49. Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. *American journal of physical anthropology*, **129**, 121-131.
50. Feng, Z.-P. and Zhang, C.-T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of protein chemistry*, **19**, 269-275.
51. 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.
52. 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*, btu083.
53. Lin, H., Deng, E.-Z., Ding, H., Chen, W. and Chou, K.-C. (2014) iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition. *Nucleic acids research*, **42**, 12961-12972.
54. Liu, B., Zhang, D., Xu, R., Xu, J., Wang, X., Chen, Q., Dong, Q. and Chou, K.-C. (2014) Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. *Bioinformatics*, **30**, 472-479.
55. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, L.S., Tacker, M. and Schuster, P. (1994) Fast Folding and Comparison of Rna Secondary Structures. *Monatsh Chem*, **125**, 167-188.
56. 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 research*, **36**, 3025-3030.
57. Xue, C., Li, F., He, T., Liu, G.-P., Li, Y. and Zhang, X. (2005) Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC bioinformatics*, **6**, 1.
58. Liu, B., Fang, L., Liu, F., Wang, X. and Chou, K.-C. (2016) iMiRNA-PseDPC: microRNA precursor identification with a pseudo distance-pair composition approach. *Journal of Biomolecular Structure and Dynamics*, **34**, 223-235.
59. Wang, J.T.L., Ma, Q., Shasha, D. and Wu, C.H. (2001) New techniques for extracting features from protein sequences. *Ibm Syst J*, **40**, 426-441.
60. White G , S.W. (1998) Using a neural network to backtranslate amino acid sequences. . *Electron. J. Biotechnol.* , 17-18.
61. Lin, K., May, A.C.W. 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.
62. 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-D205.
63. Cuff, J.A. and Barton, G.J. (2000) Application of Multiple Sequence Alignment Profiles to Improve Protein Secondary Structure Prediction. *Proteins: Structure, Function, and Bioinformatics*, **40**, 502-511.
64. Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, A., Sharma, A., Wang, J., Sattar, A., Yang, Y. and Zhou, Y. (2015) Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Scientific reports*, **5**, 11476.
65. MO., D. (1978;) A model of evolutionary change in proteins. . *Atlas Protein Seq. Struct.* , **5**, 89–99.
66. Henikoff, S. and Henikoff, J.G. (1992) Amino-Acid Substitution Matrices from Protein Blocks. *P Natl Acad Sci USA*, **89**, 10915-10919.
67. Altschul, S.F. and Koonin, E.V. (1998) Iterated profile searches with PSI-BLAST - a tool for discovery in protein databases. *Trends Biochem Sci*, **23**, 444-447.
68. Liu, B., Zhang, D.Y., Xu, R.F., Xu, J.H., Wang, X.L., Chen, Q.C., Dong, Q.W. and Chou, K.C. (2014) Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. *Bioinformatics*, **30**, 472-479.
69. Glaser, F., Rosenberg, Y., Kessel, A., Pupko, T. and Ben-Tal, N. (2005) The ConSurf-HSSP Database: The Mapping of Evolutionary Conservation Among Homologs Onto PDB Structures.

Proteins: Structure, Function, and Bioinformatics, **58**, 610-617.
70. Liu, B., Wang, X., Lin, L., Dong, Q. and Wang, X. (2008) A discriminative method for protein remote homology detection and fold recognition combining Top-n-grams and latent semantic analysis. *BMC bioinformatics*, **9**, 1.
71. Liu, B., Zhang, D., Xu, R., Xu, J., Wang, X., Chen, Q., Dong, Q. and Chou, K.C. (2014) Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. *Bioinformatics*, **30**, 472-479.
72. Liu, B., Xu, J., Lan, X., Xu, R., Zhou, J., Wang, X. and Chou, K.-C. (2014) iDNA-Prot| dis: identifying DNA-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition. *PloS one*, **9**, e106691.
73. Liu, B., Wang, X., Chen, Q., Dong, Q. and Lan, X. (2012) Using amino acid physicochemical distance transformation for fast protein remote homology detection. *PLoS One*, **7**, e46633.
74. Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, **43**, 246-255.
75. Chou, K.-C. (2005) Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics*, **21**, 10-19.
76. Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T. and Kanehisa, M. (2008) AAindex: amino acid index database, progress report 2008. *Nucleic acids research*, **36**, D202-D205.
77. Xu, R., Zhou, J., Wang, H., He, Y., Wang, X. and Liu, B. (2015) Identifying DNA-binding proteins by combining support vector machine and PSSM distance transformation. *BMC Systems Biology*, **9**, S10.
78. Zhou, J., Lu, Q., Xu, R., He, Y. and Wang, H. (2017) EL_PSSM-RT: DNA-binding residue prediction by integrating ensemble learning with PSSM Relation Transformation. *BMC bioinformatics*, **18**, 379.