Manual of Pse-Analysis

Name: Pse-Analysis

Type: Python Package

Version: 1.0

Date: 2016-12-02

Description: a Python package for DNA/RNA and protein/peptide sequence analysis based on pseudo components and kernel methods

Home-page: http://bioinformatics.hitsz.edu.cn/Pse-Analysis/

License: BSD

Download-URL: http://bioinformatics.hitsz.edu.cn/Pse-Analysis/download

Platform: MS Windows, Unix/Linux

Content

1. Introduction

Pse-Analysis is a powerful Python package for constructing machine-learning-based computational predictors for computational proteomics, genomics. Pse-Analysis is developed based on the framework of LIBSVM (Chang and Lin, 2011), and provides comprehensive functions, including feature extractions for DNA, RNA ,and proteins, parameter optimization, training SVM model, prediction, result evaluation, etc. All these functions can be easily used by the users with only two Python scripts: “train.py” and “predict.py”. Furthermore, the multiprocessing technique is applied to significantly reduce the computational cost of Pse-Analysis.
The flowchart of Pse-Analysis is shown in **Fig. 1**. More details will be introduced in the following sections.

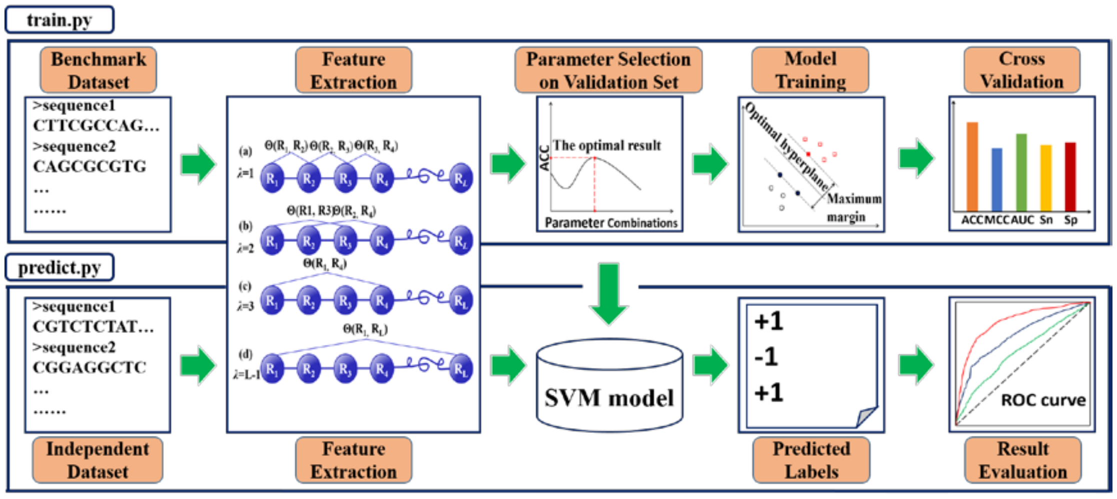

Fig. 1. The flowchart of Pse-Analysis Python package.

2. Installation

The Pse-Analysis Python package supports the Linux (64-bit) and Windows (64-bit) operating system. The full package and documents of Pse-Analysis are available at http://bioinformatics.hitsz.edu.cn/Pse-Analysis/. Before using Pse-Analysis, 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.

For Windows operating system, the Windows 7 or later versions are supported. Extract the package to a directory. After un-zip the downloaded Pse-Analysis package, make sure that the “libsvm.dll” is available in the directory “…\libsvm\windows”.

For Linux operating system, the LIBSVM should be configured firstly. Un-zip the Pse-Analysis package to a folder, for example, “~/usr”. Navigate to “~/usr/Pse-Analysis/libsvm” directory, and type the command:

```
> make
```

After executing successfully, then navigate to “~/usr/Pse-Analysis/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
```

Now, Pse-Analysis is ready to use.
For users’ convenience, an Ubuntu virtual machine image is available at http://bioinformatics.hitsz.edu.cn/Pse-Analysis/download. The Pse-Analysis package and the related system environment were all set in this image.

3. Tutorial

A simple tutorial is provided for quick start.
Task: Reconstructing the predictor iNuc-PseKNC for predicting nucleosome positioning in *C. elegans* genome, and reproducing the reported experiments (Guo, et al., 2014) by using the Pse-Analysis package.
The benchmark dataset (Guo, et al., 2014) used in this tutorial is consist of nucleosomal and linker sequences from *C. elegans*. There are 2567 positive samples and 2608 negative samples, which can be accessed from http://lin.uestc.edu.cn/server/iNucPseKNC/data/Supp-2.pdf. In this tutorial, the “DNA_pos.txt” represents the positive dataset and the “DNA_neg.txt” represents the negative dataset. Both the two files are given in the “/data/example” directory.

The predictor iNuc-PseKNC can be constructed based on the benchmark dataset by using the following command line:

```
python train.py ./data/example/DNA_pos.txt ./data/example/DNA_neg.txt DNA -m dna.model -v 10
```

Firstly, the input positive and negative datasets of DNA sequences are converted into fixed length feature vectors by using the PseKNC (Guo, et al., 2014; Lin, et al., 2014). Then the parameters of the feature extraction algorithm PseKNC are automatically optimized on the validation sets. The process of parameter selection and the optimal parameters will be output on the screen:

```
Processing...
Parameter selection is in processing...


Iteration   k =   2   lamada =   5    finished.
Iteration   k =   2   lamada =   7    finished.
Iteration   k =   3   lamada =   5    finished.
Iteration   k =   2   lamada =   9    finished.
Iteration   k =   2   lamada =   11   finished.
Iteration    k =    3    lamada =    7    finished.
Iteration    k =    3    lamada =    9    finished.
............
............
Iteration    k =    6    lamada =    13    finished.
Iteration    k =    6    lamada =    11    finished.
Iteration    k =    6    lamada =    15    finished.
The time cost for parameter selection is 744.72s
Parameter selection completed.
The optimal parameters for the dataset are: k = 3    lambda = 5
```

The next step is model training. The feature vectors generated by feature extraction algorithm with optimized parameters are fed into the Support Vector Machines (SVMs) to train the classifiers. A SVM model will be generated and will be saved as a file along with optimal parameters. In order to evaluate the performance of the model, 10-fold cross validation is used and Five commonly used performance measures are calculated, including accuracy (ACC), Mathew's Correlation Coefficient (MCC), Area Under roc Curve (AUC), sensitivity (Sn), and specificity (Sp). Furthermore, the corresponding ROC curve is also provided and saved in a PNG file. The related information will be output on the screen:

```
Model training is in processing...
The cross validation results are as follows:
ACC = 0.8342
MCC = 0.6759
AUC = 0.9050
Sn    = 0.9060
Sp    = 0.7629
The ROC curve has been saved. You can check it here:
/Pse-Analysis/data/final_results/cross_validation.png

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

Done.
Used time: 789.19s
```

The generated ROC curve is shown in **Fig. 2**.
The whole process of the command for analyzing *elegans* genome dataset is shown above. For more details of the process please refer to the following sections.

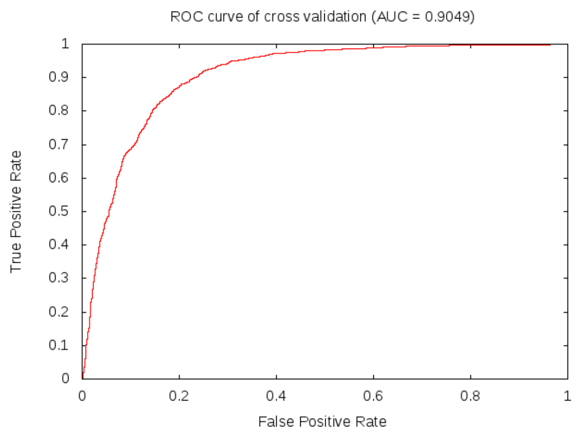

Fig .2. The ROC curve of cross validation.

4. Function description

4.1 Directory structure

The main directory contains several Python files and folders. “train.py” and “predict.py” are two executive scripts. 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. “pse.py” implements the methods used for feature extraction, including Pseudo k-tuple nucleotide composition (PseKNC) (Guo, et al., 2014; Lin, et al., 2014), General parallel correlation pseudo dinucleotide composition (PC-PseDNC) (Chen, et al., 2015), and Parallel correlation pseudo amino acid composition (PC-PseAAC) (Chou, 2001). “libsvm” folder contains the LIBSVM package. The tool for drawing ROC curve is in the “gnuplot” folder. “docs” folder contains the manual of Pse-Analysis. In “data” folder, there are four subfolders: “example” folder contains the dataset files used in the example, a shell script file and a batch file used for example demonstration on Linux and Windows operating systems respectively; “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 data files used for feature extraction are stored in the “pse_data” folder. Modifications of these files are not suggested.

4.2 train.py

Basic functions

The “train.py” is used for training SVM-based predictors and evaluating their performance based on the input benchmark datasets. There are four main processes of “train.py”, including feature extraction, parameter selection, model training and cross validation. In the feature extraction process, the features of DNA, RNA, and protein sequences can be extracted by Pseudo k-tuple nucleotide composition (PseKNC) (Guo, et al., 2014; Lin, et al., 2014), General parallel correlation pseudo dinucleotide composition (PC-PseDNC) (Chen, et al., 2015), and Parallel correlation pseudo amino acid composition (PC-PseAAC) (Chou, 2001), respectively. During the parameter selection process, the parameters in the feature extraction algorithm 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, the LIBSVM package is employed to train the prediction models. Finally, in the cross validation process, the performance of the constructed predictors is evaluated by 5-fold cross-validation. For more details of these four processes, please refer to the "**Methods description**" section.

Input and output

The input files of "train.py" are two FASTA files, storing the positive samples, and negative samples, respectively. The output file is the trained SVM model listing the parameters used in the training process and the log information, for example:

```
k,4,lamada,6,w,0.5,c,128,g,0.5,b,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 ……
……
```

4.3 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". The performance of the constructed predictors is evaluated by five common performance measures, and the corresponding ROC curves can also be generated. 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 dataset in FASTA format. 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.

5. Commands

5.1 “train.py” Usage

Command line arguments for “train.py”:

required	**description**
Posfile	The input positive sequence file in FASTA format.
Negfile	The input negative sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
-m M	The name of the trained SVM model.

Optional	**description**
-h, --help	Show this help message and exit.
-k K	The k value of kmer. It works only with PseKNC method for DNA sequences.
-w W	The value of the parameter *w* for PseKNC, PC-PseDNC, and PC-PseAAC. Default value is 1.0.
-lamada LAMADA	The value of parameter lambda for for PseKNC, PC-PseDNC, and PC-PseAAC.
-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. i: (character 'i') independent test set method.
-ipos IPOS	The name of positive independent test dataset. If the parameter “-v” is specified as “i”, a positive or negative independent test dataset should be included.
-ineg INEG	The name of negative independent test dataset. If the parameter “-v” is specified as “i”, a positive or negative independent test dataset should be included.
-c C	The parameter C of RBF kernel. Default value is 128.
-g G	The parameter gamma of RBF kernel. Default value is 0.5.
-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 the number of all available CPU cores.

5.2 “predict.py” Usage

Command line arguments for “predict.py”:

required	description
inputfile	The input sequence file in FASTA format.
{DNA, RNA, Protein}	The sequence type.
-m M	The name of trained model generated by using “train.py”.

optional	description
-h, --help	Show this help message and exit.
-labels LABELS	The real label file. Optional.

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

5.3 Example

An example of using Pse-Analysis to construct machine learning predictor for solving a specific task in bioinformatics is given.
Example: Reconstructing the predictor PseDNA-Pro for DNA binding protein identification based on the a benchmark dataset (Liu, et al., 2015), and evaluating its performance on an independent dataset (Lou, et al., 2014) by using Pse-Analysis package.

The benchmark (Liu, et al., 2015) dataset contains 525 positive samples and 550 negative samples. There are 82 positive samples and 99 negative samples in the independent dataset. The benchmark dataset and independent dataset are available at http://bioinformatics.hitsz.edu.cn/PseDNA-Pro/Resources/benchmark_dataset.pdf, and, http://journals.plos.org/plosone/article/asset?unique&id=info:doi/10.1371/journal.pone.0086703.s002, 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.

The predictor PseDNA-Pro can be constructed based on the benchmark dataset by using the following common line:

```
python train.py ./data/example/protein_pos.txt ./data/example/protein_neg.txt Protein -m
protein.model -v 10
```

The output information is as follows:

```
Processing...
Parameter selection is in processing...

Iteration   w =   0.0   lamada =   5   finished.
Iteration   w =   1.0   lamada =   5   finished.
Iteration   w =   0.8   lamada =   5   finished.
Iteration   w =   0.4   lamada =   5   finished.
Iteration   w =   0.0   lamada =   7   finished.
Iteration   w =   0.6   lamada =   5   finished.
Iteration   w =   0.2   lamada =   5   finished.
Iteration   w =   0.4   lamada =   7   finished.
Iteration   w =   0.4   lamada =   9   finished.
Iteration   w =   0.6   lamada =   7   finished.
Iteration   w =   0.8   lamada =   9   finished.
Iteration   w =   0.8   lamada =   7   finished.
Iteration   w =   0.2   lamada =   7   finished.
Iteration   w =   1.0   lamada =   7   finished.
Iteration   w =   0.2   lamada =   9   finished.
Iteration   w =   0.0   lamada =   9   finished.
Iteration   w =   0.0   lamada =   11   finished.
Iteration   w =   0.6   lamada =   9   finished.
Iteration   w =   0.2   lamada =   11   finished.
Iteration   w =   1.0   lamada =   11   finished.
Iteration   w =   1.0   lamada =   9   finished.
Iteration   w =   0.4   lamada =   11   finished.
Iteration   w =   0.0   lamada =   13   finished.
Iteration   w =   0.4   lamada =   13   finished.
Iteration   w =   0.2   lamada =   13   finished.
Iteration   w =   0.8   lamada =   11   finished.
Iteration   w =   0.6   lamada =   15   finished.
Iteration   w =   0.8   lamada =   13   finished.
Iteration   w =   0.6   lamada =   13   finished.
Iteration   w =   1.0   lamada =   13   finished.
Iteration   w =   0.6   lamada =   11   finished.
Iteration   w =   0.8   lamada =   15   finished.
Iteration   w =   0.2   lamada =   15   finished.
Iteration   w =   0.0   lamada =   15   finished.
Iteration   w =   1.0   lamada =   15   finished.
Iteration   w =   0.4   lamada =   15   finished.
The time cost for parameter selection is 9.30s
Parameter selection completed.
The optimal parameter for the dataset is: w = 0.0   lambda = 7

Model training is in processing...
The cross validation results are as follows:
ACC = 0.7524
MCC = 0.5028
AUC = 0.8128
Sn   = 0.7387
Sp   = 0.7651
The ROC curve has been saved. You can check it here:
/Pse-Analysis/data/final_results/cross_validation.png
```

```
Model training completed.
The model has been saved. You can check it here:
/Pse-Analysis/data/final_results/protein.model
Done.
Used time: 10.48s
```

Then, the performance of PseDNA-Pro can be further evaluated by using the independent dataset:

```
python predict.py ./data/example/protein_test.txt Protein -labels ./data/example/labels.txt -m
protein.model
```

The output information is as follows:

```
Processing...
The parameters of feature extraction method:
w = 0.0    lambda = 7
The parameters of RBF kernel:
c = 128    g = 0.5
The performance evaluations are as follows:

ACC = 0.6774
MCC = 0.3578
AUC = 0.7283
Sn    = 0.7419
Sp    = 0.6129

The ROC curve has been saved. You can check it here:
/Pse-Analysis/data/final_results/predicted_roc.png

The predicted labels have been saved. You can check it here:
/Pse-Analysis/data/final_results/output_labels.txt

Done.
Used time: 0.79s
```

As shown in this example, the PseDNA-Pro can be easily constructed based on the benchmark dataset by using the script “train.py”, and then evaluated on the independent dataset by using “predict.py”.

One shell script file and one batch file used for example demonstration have been provided for users. The “demo.sh” and “demo.bat” are for Linux and Windows operating systems, respectively, which are accessible in “/data/example” folder.

6. Methods description

6.1 Feature extraction

In the Pse-Analysis, three state-of-the-art algorithms are employed, including Pseudo k-tuple nucleotide composition (PseKNC) (Guo, et al., 2014; Lin, et al., 2014), Parallel correlation pseudo dinucleotide composition (PC-PseDNC) (Chen, et al., 2015), and Parallel correlation pseudo amino acid composition (PC-PseAAC) (Chou, 2001) for extracting the features of DNA, RNA, and protein sequences, respectively. The detailed information of these three approaches will be introduced in the following sections.

6.1.1 Pseudo k-tuple nucleotide composition (PseKNC)

Suppose a DNA sequence **D** with L nucleic acid residues; i.e.

$$\mathbf{D} = \mathrm{R}_1\mathrm{R}_2\mathrm{R}_3\mathrm{R}_4\mathrm{R}_5\mathrm{R}_6\mathrm{R}_7 \cdots \mathrm{R}_L \tag{1}$$

where R_1 represents the nucleic acid residue at the sequence position 1, R_2 the nucleic acid residue at position 2 and so forth.

The feature vector of **D** is defined:

$$\mathbf{D} = [d_1\ d_2 \cdots d_{4^k}\ d_{4^k+1} \cdots d_{4^k+\lambda}]^{\mathrm{T}} \tag{2}$$

where

$$d_u = \begin{cases} \dfrac{f_u}{\sum_{i=1}^{4^k} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (1 \le u \le 4^k) \\[2ex] \dfrac{w\theta_{u-4^k}}{\sum_{i=1}^{4^k} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (4^k \le u \le 4^k + \lambda) \end{cases} \tag{3}$$

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

$$\theta_j = \frac{1}{L\text{-}j\text{-}1}\sum_{i=1}^{L-j-1}\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+j}\mathrm{R}_{i+j+1}) \quad (j = 1, 2, \cdots, \lambda; \lambda < L) \tag{4}$$

which represents the j-tier structural correlation factor between all the j-th most contiguous dinucleotides. The correlation function $\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+j}\mathrm{R}_{i+j+1})$ is defined by

$$\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+j}\mathrm{R}_{i+j+1}) = \frac{1}{\mu}\sum_{v=1}^{\mu}[P_v(\mathrm{R}_i\mathrm{R}_{i+1}) - P_v(\mathrm{R}_{i+j}\mathrm{R}_{i+j+1})]^2 \tag{5}$$

where μ is the number of physicochemical indices, in this study, 6 indices reflecting the local DNA structural properties (**Table 1**) are employed to generate the PseKNC feature vector; $P_v(\mathrm{R}_i\mathrm{R}_{i+1})$ ($P_v(\mathrm{R}_{i+j}\mathrm{R}_{i+j+1})$) represents the numerical value of the *v*-th $(v = 1, 2, \cdots, \mu)$ physicochemical index for the dinucleotide $\mathrm{R}_i\mathrm{R}_{i+1}$ ($\mathrm{R}_{i+j}\mathrm{R}_{i+j+1}$) at position *i* (*i*+*j*).

Before substituting the values of physicochemical index into **Eq. 5**, they should be normalized by following equation:

$$P_v(\mathrm{R}_i\mathrm{R}_{i+1}) = \frac{P_v(\mathrm{R}_i\mathrm{R}_{i+1}) - \langle P_v(\mathrm{R}_i\mathrm{R}_{i+1})\rangle}{\mathrm{SD}\langle P_v(\mathrm{R}_i\mathrm{R}_{i+1})\rangle} \tag{6}$$

where the symbol < > means taking the average of the quantity therein over the 16 different combinations of A, C, G, T for $\mathrm{R}_i\mathrm{R}_{i+1}$, and SD means the corresponding standard deviation.

Table 1. The values of the 6 physicochemical indices for dinucleotides (DNA).

Physicochemical properties	**Rise (DNA)**	**Roll (DNA)**	**Shift (DNA)**	**Slide (DNA)**	**Tilt (DNA)**	**Twist (DNA)**
AA	7.650	2.260	1.690	0.026	0.020	0.038
AC	8.930	3.030	1.320	0.036	0.023	0.038
AG	7.080	2.030	1.460	0.031	0.019	0.037
AT	9.070	3.830	1.030	0.033	0.022	0.036
CA	6.380	1.780	1.070	0.016	0.017	0.025
CC	8.040	1.650	1.430	0.026	0.019	0.042
CG	6.230	2.000	1.080	0.014	0.016	0.026
CT	7.080	2.030	1.460	0.031	0.019	0.037
GA	8.560	1.930	1.320	0.025	0.020	0.038
GC	9.530	2.610	1.200	0.025	0.026	0.036
GG	8.040	1.650	1.430	0.026	0.019	0.042
GT	8.930	3.030	1.320	0.036	0.023	0.038
TA	6.230	1.200	0.720	0.017	0.016	0.018
TC	8.560	1.930	1.320	0.025	0.020	0.038
TG	6.380	1.780	1.070	0.016	0.017	0.025
TT	7.650	2.260	1.690	0.026	0.020	0.038

6.1.2 General parallel correlation pseudo dinucleotide composition (PC-PseDNC)

Suppose a RNA sequence **R** with *L* nucleic acid residues; i.e.

$$\mathbf{R} = \mathrm{R}_1\mathrm{R}_2\mathrm{R}_3\mathrm{R}_4\mathrm{R}_5\mathrm{R}_6\mathrm{R}_7 \cdots \mathrm{R}_L \tag{7}$$

where R_1 represents the nucleic acid residue at the sequence position 1, R_2 the nucleic acid residue at position 2 and so forth.

The PC-PseDNC (Chen, et al., 2015) feature vector of **R** is defined:

$$\mathbf{R} = [d_1\ d_2\ \cdots\ d_{16}\ d_{16+1}\ \cdots\ d_{16+\lambda}]^{\mathrm{T}} \tag{8}$$

where

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (1 \le k \le 16) \\ \dfrac{w\theta_{k-16}}{\sum_{i=1}^{16} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (16+1 \le k \le 16+\lambda) \end{cases} \tag{9}$$

where f_k（k=1,2,⋯,16) is the normalized occurrence frequency of dinucleotide in the RNA sequence; the parameter λ is an integer, representing the highest counted rank (or tier) of the correlation along a RNA sequence; w is the weight factor ranging from 0 to 1; $\theta_j (j = 1, 2, \cdots, \lambda)$ is called the j-tier correlation factor reflecting the sequence-order correlation between all the i-th most contiguous dinucleotides along an RNA sequence, which is defined:

$$\begin{cases} \theta_1 = \dfrac{1}{L\text{-}2}\sum\limits_{i=1}^{L-2}\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+1}\mathrm{R}_{i+2}) \\ \theta_2 = \dfrac{1}{L\text{-}3}\sum\limits_{i=1}^{L-3}\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+2}\mathrm{R}_{i+3}) \\ \theta_3 = \dfrac{1}{L\text{-}4}\sum\limits_{i=1}^{L-4}\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+3}\mathrm{R}_{i+4}) & (\lambda < L) \\ \ldots\ldots \\ \theta_\lambda = \dfrac{1}{L\text{-}1\text{-}\lambda}\sum\limits_{i=1}^{L-1-\lambda}\Theta(\mathrm{R}_i\mathrm{R}_{i+1}, \mathrm{R}_{i+\lambda}\mathrm{R}_{i+\lambda+1}) \end{cases} \tag{10}$$

where the correlation function is given by

$$\Theta(\mathrm{R}_i\mathrm{R}_{i+1}\mathrm{R}_j\mathrm{R}_{j+1}) = \frac{1}{\mu}\sum_{\mu-1}^{\mu}[P_\mu(\mathrm{R}_i\mathrm{R}_{i+1}) - P_\mu(\mathrm{R}_j\mathrm{R}_{j+1})]^2 \tag{11}$$

where μ is the number of physicochemical indices considered that are listed in the **Table 2**; Here in Pse-Analysis, we use 6 physicochemical indices: Rise (RNA), Roll (RNA), Shift (RNA), Slide (RNA), Tilt (RNA), Twist (RNA). $P_u(\mathrm{R}_i\mathrm{R}_{i+1})$ $(P_u(\mathrm{R}_j\mathrm{R}_{j+1}))$ represents the numerical value of the u-th $(u = 1, 2, \cdots, \mu)$ physicochemical index for the dinucleotide $\mathrm{R}_i\mathrm{R}_{i+1}(\mathrm{R}_j\mathrm{R}_{j+1})$ at position $i(j)$.

Before substituting the values of physicochemical index into **Eq. 11**, they should be normalized by following equation:

$$P_\mu(\mathrm{R}_i\mathrm{R}_{i+1}) = \frac{P_\mu(\mathrm{R}_i\mathrm{R}_{i+1}) - \langle P_\mu(\mathrm{R}_i\mathrm{R}_{i+1})\rangle}{\mathrm{SD}\langle P_\mu(\mathrm{R}_i\mathrm{R}_{i+1})\rangle} \tag{12}$$

where the symbol < > means taking the average of the quantity therein over the 16 different combinations of A, C, G, U for R_iR_{i+1}, and SD means the corresponding standard deviation.

Table 2. The values of the 6 physicochemical indices for dinucleotides (RNA).

Physicochemical properties	**Rise (RNA)**	**Roll (RNA)**	**Shift (RNA)**	**Slide (RNA)**	**Tilt (RNA)**	**Twist (RNA)**
AA	3.180	7.000	-0.080	-1.270	-0.800	31.000
AC	3.240	4.800	0.230	-1.430	0.800	32.000
AG	3.300	8.500	-0.040	-1.500	0.500	30.000
AU	3.240	7.100	-0.060	-1.360	1.100	33.000
CA	3.090	9.900	0.110	-1.460	1.000	31.000
CC	3.320	8.700	-0.010	-1.780	0.300	32.000
CG	3.300	12.100	0.300	-1.890	-0.100	27.000
CU	3.300	8.500	-0.040	-1.500	0.500	30.000
GA	3.380	9.400	0.070	-1.700	1.300	32.000
GC	3.220	6.100	0.070	-1.390	0.000	35.000
GG	3.320	12.100	-0.010	-1.780	0.300	32.000
GU	3.240	4.800	0.230	-1.430	0.800	32.000
UA	3.260	10.700	-0.020	-1.450	-0.200	32.000
UC	3.380	9.400	0.070	-1.700	1.300	32.000
UG	3.090	9.900	0.110	-1.460	1.000	31.000
UU	3.180	7.000	-0.080	-1.270	-0.800	31.000

6.1.3 Parallel correlation pseudo amino acid composition (PC-PseAAC)

PC-PseAAC (Chou, 2001) is an approach incorporating the contiguous local sequence-order information and the global sequence-order information into the feature vector of the protein sequence.
Suppose a protein sequence **P** with *L* amino acid residues; i.e.

$$\mathbf{P} = R_1R_2R_3R_4R_5R_6R_7 \cdots R_L \tag{13}$$

where R_1 represents the amino acid residue at the sequence position 1, R_2 the amino acid residue at position 2 and so forth. The PC-PseAAC (Chou, 2001) feature vector of **P** is defined:

$$\mathbf{P} = [x_1 \ x_2 \ \cdots \ x_{20} \ x_{20+1} \ \cdots \ x_{20+\lambda}]^{\mathrm{T}} \tag{14}$$

where

$$x_u = \begin{cases} \dfrac{f_u}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (1 \le u \le 20) \\ \dfrac{w\theta_{u-20}}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{\lambda}\theta_j} & (20+1 \le u \le 20+\lambda) \end{cases} \tag{15}$$

where f_i (i=1,2,···,20) is the normalized occurrence frequency of the 20 amino acids in the protein **P**; the parameter λ is an integer, representing the highest counted rank (or tier) of the correlation along a protein sequence; w is the weight factor ranging from 0 to 1; $\theta_j (j = 1,2,\cdots,\lambda)$ is called the j-tier correlation factor reflecting the sequence-order correlation between all the j-th most contiguous residues along a protein chain, which is defined:

$$\begin{cases} \theta_1 = \dfrac{1}{L\text{-}1}\sum_{i=1}^{L-1}\Theta(\mathrm{R}_i, \mathrm{R}_{i+1}) \\ \theta_2 = \dfrac{1}{L\text{-}2}\sum_{i=1}^{L-2}\Theta(\mathrm{R}_i, \mathrm{R}_{i+2}) \\ \theta_3 = \dfrac{1}{L\text{-}3}\sum_{i=1}^{L-3}\Theta(\mathrm{R}_i, \mathrm{R}_{i+3}) \\ \cdots\cdots \\ \theta_\lambda = \dfrac{1}{L\text{-}\lambda}\sum_{i=1}^{L-\lambda}\Theta(\mathrm{R}_i, \mathrm{R}_{i+\lambda}) \end{cases} \quad (\lambda < L) \tag{16}$$

where the correlation function is given by

$$\Theta(\mathrm{R}_i, \mathrm{R}_j) = \frac{1}{3}\left\{\left[H_1(\mathrm{R}_j) - H_1(\mathrm{R}_i)\right]^2 + \left[H_2(\mathrm{R}_j) - H_2(\mathrm{R}_i)\right]^2 + \left[M(\mathrm{R}_j) - M(\mathrm{R}_i)\right]^2\right\} \tag{17}$$

where $H_1(\mathrm{R}_i)$, $H_2(\mathrm{R}_i)$, and $M(\mathrm{R}_i)$ are, respectively, the hydrophobicity value, hydrophilicity value, and side-chain mass (**Table 3**) of the amino acid R_i; Note that before substituting the values of hydrophobicity, hydrophilicity, and side-chain mass into **Eq. 17**, they are all subjected to a standard conversion as described by the following equation:

$$\begin{cases} H_1(i) = \dfrac{H_1^0(i) - \sum_{i-1}^{20}\dfrac{H_1^0(i)}{20}}{\sqrt{\dfrac{\sum_{i-1}^{20}[H_1^0(i) - \sum_{i-1}^{20}\dfrac{H_1^0(i)}{20}]^2}{20}}} \\ H_2(i) = \dfrac{H_2^0(i) - \sum_{i-1}^{20}\dfrac{H_2^0(i)}{20}}{\sqrt{\dfrac{\sum_{i-1}^{20}[H_2^0(i) - \sum_{i-1}^{20}\dfrac{H_2^0(i)}{20}]^2}{20}}} \\ M(i) = \dfrac{M^0(i) - \sum_{i-1}^{20}\dfrac{M^0(i)}{20}}{\sqrt{\dfrac{\sum_{i-1}^{20}[M^0(i) - \sum_{i-1}^{20}\dfrac{M^0(i)}{20}]^2}{20}}} \end{cases} \tag{18}$$

where $H_1^0(i)$ is the original hydrophobicity value of the *i*-th amino acid; $H_2^0(i)$ the corresponding original hydrophilicity value; $M^0(i)$ the mass of the *i*-th amino acid side chain.

Table 3. The values of the 3 physicochemical indices for amino acids.

Physicochemical properties	**hydrophobicity**	**hydrophilicity**	**mass**
A	0.620	-0.500	71.079
R	-2.530	3.000	156.188
N	-0.780	0.200	114.104
D	-0.090	3.000	115.086
C	0.290	-1.000	103.145
Q	-0.850	0.200	128.131
E	-0.740	3.000	129.116
G	0.480	0.000	57.052
H	-0.400	-0.500	137.141
I	1.380	-1.800	113.160
L	1.530	-1.800	113.160
K	-1.500	3.000	128.170
M	0.640	-1.300	131.990
F	1.190	-2.500	147.177
P	0.120	0.000	97.177
S	-0.180	0.300	87.078
T	-0.050	-0.400	101.105
W	0.810	-3.400	186.123
Y	0.260	-2.300	163.176
V	1.800	-1.500	99.133

6.2 Parameter selection

As introduced above, there are several parameters in the three feature extraction methods, which should be optimized when constructing a predictor. Pse-Analysis is able to automatically optimize these parameters based on the best performance on the validation sets. To improve the efficiency of this procedure, multiprocessing technique is applied, which significantly reduces the computational cost.

For PseKNC, two parameters *k* and λ are optimized. The ranges of parameters are as follows:

$$\begin{cases} 2 \le k \le 6, \text{ step } \Delta=1 \\ 5 \le \lambda \le 15, \text{ step } \Delta=2 \end{cases} \qquad (19)$$

For PC-PseDNC, *w* and *λ* are the parameters to be optimized. The ranges of parameters are as follows:

$$\begin{cases} 0 \le w \le 1, \text{ step } \Delta=0.2 \\ 5 \le \lambda \le 15, \text{ step } \Delta=2 \end{cases} \qquad (20)$$

For PC-PseAAC, *w* and *λ* are the parameters to be optimized. The ranges of parameters are as follows:

$$\begin{cases} 0 \le w \le 1, \text{ step } \Delta=0.2 \\ 5 \le \lambda \le 15, \text{ step } \Delta=2 \end{cases} \qquad (21)$$

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.

6.3 Model training

In the model training process, this model is trained based on LIBSVM with RBF kernel.
The trained SVM model and all the parameters are saved in a separate file, which will be used as the input for "predict.py".

6.4 Cross validation

Pse-Analysis 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.

The performance 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) (Fawcett, 2006) curve will also be generated and saved in a PNG file.

6.5 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, calculate the five performance measures (Acc, MCC, AUC, Sn, and Sp), and generate the corresponding ROC curve saved as a PNG file.

References

Chang, C.C. and Lin, C.J. (2011) LIBSVM: A Library for Support Vector Machines, *Acm T Intel Syst Tec*, **2**, 1-27.

Chen, W., *et al.* (2015) PseKNC-General: a cross-platform package for generating various modes of pseudo nucleotide compositions, *Bioinformatics*, **31**, 119-120.

Chou, K.C. (2001) Prediction of protein cellular attributes using pseudo-amino acid composition, *Proteins: Structure, Function, and Bioinformatics*, **43**, 246-255.

Fawcett, T. (2006) An introduction to ROC analysis, *Pattern recognition letters*, **27**, 861-874.

Guo, S.-H., *et al.* (2014) iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition, *Bioinformatics*, btu083.

Lin, H., *et al.* (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.

Liu, B., *et al.* (2015) PseDNA-Pro: DNA-Binding Protein Identification by Combining Chou's PseAAC and Physicochemical Distance Transformation, *Molecular Informatics*, **34**, 8-17.

Lou, W., *et al.* (2014) Sequence based prediction of DNA-binding proteins based on hybrid feature selection using random forest and Gaussian naive Bayes, *PLoS One*, **9**, e86703.