SeqFea-Learn: A Python pipeline tool for feature extraction, feature selection, machine learning and deep learning based on DNA, RNA and protein sequence data

Name Surname[1,*], Name Surname[2] and Name Surname[2]

[1] Department, Institution, Town, State, Postcode, Country
[2] Department, Institution, Town, State, Postcode, Country

Methods Description

CONTENTS

Introduction

SeqFea-Learn is a Python pipeline tool for analyzing DNA, RNA, and protein sequencing data that integrated 19 feature extraction methods for DNA, 15 feature extraction methods for RNA, 32 feature extraction methods for Protein sequencing data, 21 feature selection methods, 15 dimensionality reduction methods, 7 clustering methods, 5 sampling methods, 10 classification methods, and 3 deep learning methods. This document will provide user detailed descriptions for each method.

DNA feature extraction method

SeqFea-Learn contains 19 DNA feature extraction methods.

Feature Extraction Method	Method Number
Kmer	1
Reverse compliment kmer (RCKmer)	2
Nucleic acid composition (NAC)	3
Di-nucleotide composition (DNC)	4
Tri-Nucleotide Composition (TNC)	5
Binary encoding (BE)	6
zCurve Mathematical Formula (zCurve)	7
Dinucleotide based auto covariance (DAC)	8
Dinucleotide based cross covariance (DCC)	9
Di-nucleotide based auto-cross covariance (DACC)	10
Tri-nucleotide based auto covariance (TAC)	11
Tri-nucleotide based cross covariance (TCC)	12
Tri-nucleotide based auto-cross covariance (TACC)	13
Position-specific dinucleotide propensity (PSDNP)	14
Position-specific trinucleotide propensity (PSTNP)	15
MonoKGap theoretical description (MonoKGap)	16
MonoDiKGap theoretical description (MonodiKGap)	17
Pseudo di-nucleotide composition (PseDNC)	18
Pseudo k-tuple nucleotide composition (PseKNC)	19

1. Kmer

For Kmer descriptor, the DNA sequences are represented as the occurrence frequencies of k neighboring nucleic acids. The Kmer (k=3) descriptor can be calculated as:

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

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

2. Reverse compliment kmer (RCKmer)

The reverse compliment kmer is a variant of kmer descriptor, in which the kmers are not expected to be strand specific. For instance, for a DNA sequence, there are 16 types of 2-mers (i.e. 'AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'CT', 'GA', 'GC', 'GG', 'GT', 'TA', 'TC', 'TG', 'TT'), 'TT' is reverse compliment with 'AA'. After removing the reverse compliment kmers, there are only 10 distinct kmers in the reverse compliment kmer approach ('AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'GA', 'GC', 'TA').

3. Pseudo dinucleotide composition (PseDNC)

PseDNC is an approach incorporating the contiguous local sequence-order information and the global sequence-order information into the feature vector of the DNA sequence. Given a DNA sequence D, the PseDNC feature vector of D is defined:

$$D = [d_1, d_2, d_3, \cdots, d_{16}, d_{16+1}, \cdots, d_{16+\lambda}]$$

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

where f_k is the normalized occurrence frequency of dinucleotides 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 ranged from 0 to 1, θ_j is called the correlation factor that reflects the sequence-order correlation between all the most contiguous dinucleotides along a RNA sequence, which is defined:

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

where the correlation function is given by

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

where μ is the number of physicochemical indices, in this approach, 6 indices reflecting the local RNA structural properties are employed to generate the PseDNC feature vector, $P_u(R_iR_{i+1}), P_u(R_jR_{j+1})$ represents the numerical value of the *u-th* physicochemical index of the dinucleotide R_iR_{i+1}, R_jR_{j+1}.

4. Pseudo k-tuple nucleotide composition (PseKNC)

PseKNC extends the PseDNC approach by incorporating *k*-tuple nucleotide composition. Given a DNA sequence D, the feature vector of D is defined:

$$D = [d_1, d_2, d_3, \cdots, d_{4^k}, d_{4^k+1}, \cdots, d_{4^k+\lambda}]$$

$$d_k = \begin{cases} \dfrac{f_k}{\sum_{i=1}^{4^k} f_i + \omega \sum_{j=1}^{\lambda} \theta_j} & (1 \le k \le 4^k) \\ \dfrac{\omega \theta_{k-4^k}}{\sum_{i=1}^{4^k} f_i + \omega \sum_{j=1}^{\lambda} \theta_j} & (4^k \le k \le 4^k + \lambda) \end{cases}$$

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

$$\theta_j = \frac{1}{L-j-1} \sum_{i=1}^{L-j-1} \Theta(R_i R_{i+1}, R_{i+j} R_{i+j+1}) \; (j = 1, 2, \cdots, \lambda; \lambda < L)$$

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

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

where μ is the number of physicochemical indices, in this study, 6 indices reflecting the local DNA structural properties are employed to generate the PseKNC feature vector; $P_v(R_i R_{i+1}), P_v(R_{i+j} R_{i+j+1})$ represents the numerical value of the v-th physicochemical index for the dinucleotide $R_i R_{i+1}, R_{i+j} R_{i+j+1}$.

5. Dinucleotide-based auto covariance (DAC)

The DAC measures the correlation of the same physicochemical index between two dinucleotides separated by a distance of lag along the sequence, which can be calculated as:

$$DAC(u, l) = \sum_{i=1}^{L-l-1} (P_u(R_i R_{i+1}) - \bar{P}_u)(P_u(R_{i+l} + R_{i+l+1}) - \bar{P}_u) / (L - l - 1)$$

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

In such a way, the length of DAC feature vector is $N \times lag$, where N is the number of physicochemical indices, and lag is the maximum of sequence.

6. Dinucleotide-based cross covariance (DCC)

Given a DNA sequence, the DCC approach measures the correlation of two different physicochemical indices between two dinucleotides separated by lag nucleic acids along the sequence, which can be calculated by:

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

where u_1,u_2 are two different physicochemical indices, L is the length of the DNA sequence, $P_{u_1}\left(R_iR_{i+1}\right),P_{u_2}\left(R_iR_{i+1}\right)$ is the numerical value of the physicochemical index u_1,u_2 for the dinucleotide R_iR_{i+1}. $\bar{P}_{u_1}$ is the average value for physicochemical index value u_1 along the whole sequence:

$$\overline{P_u}=\sum_{j=1}^{L-1}P_u\left(\mathrm{R}_j\mathrm{R}_{j+1}\right)/(L-1)$$

In such a way, the length of the DCC feature vector is $N\times(N-1)\times lag$, lag is the maximum of sequence; N is the number of physicochemical indices.

7. Dinucleotide-based auto-cross covariance (DACC)

DACC is a combination of DAC and DCC. Therefore, the length of the DACC feature vector is $N \times N \times lag$, where N is the number of physicochemical indices and lag is the maximum of sequence.

8. Trinucleotide-based auto covariance (TAC)

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

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

where u is a physicochemical index, L is the length of the DNA sequence $P_u(R_iR_{i+1}R_{i+2})$ means the numerical value of the physicochemical index u for the dinucleotide $R_iR_{i+1}R_{i+2}$ at position i, $\bar{P}_u$ is the average value for physicochemical index u along the whole sequence. The dimension of TAC feature vector is $N \times lag$, where N is the number of physicochemical indices, and lag is the maximum of sequence.

9. Trinucleotide-based cross covariance (TCC)

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

$$TCC(u_1,u_2,l) = \sum_{i=1}^{L-l-2} \left(P_{u_1}\left(R_iR_{i+1}R_{i+2}\right) - \bar{P}_{u_1}\right)\left(P_{u_2}\left(R_{i+l}R_{i+l+1}R_{i+l+2}\right) - \bar{P}_{u_2}\right) / (L-l-2)$$

where u_1, u_2 are two different physicochemical indices, L is the length of the DNA sequence, $P_{u_1}\left(R_iR_{i+1}R_{i+2}\right), P_{u_2}\left(R_iR_{i+1}R_{i+2}\right)$ is the numerical value of the physicochemical index

u_1,u_2 for the dinucleotide $R_iR_{i+1}R_{i+2}$. $\bar{P}_{u_1}$ is the average value for physicochemical index value u_1 along the whole sequence:

$$\overline{P_u} = \sum_{j=1}^{L-1} P_u\left(R_jR_{j+1}R_{j+2}\right)/(L-1)$$

In such a way, the length of the TCC feature vector is $N\times(N-2)\times lag$, lag is the maximum of sequence; N is the number of physicochemical indices.

10. Trinucleotide-based auto-cross covariance (TACC)

The trinucleotide-based auto-cross covariance (TACC) encoding is a combination of TAC and TACC encoding. Thus, the dimension of the TACC encoding is $N\times N\times lag$, where N is the number of physicochemical indices and lag is the maximum of DNA sequence.

11. Nucleic acid composition (NAC)

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

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

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

12. Di-nucleotide composition (DNC)

The Di-Nucleotide Composition gives 16 descriptors. It is defined as:

$$D(r,s)=\frac{N_{rs}}{N-1},\ r,s\in\{A,C,G,T(U)\}$$

where N_{rs} is the number of di-nucleotide represented by nucleic acid types *r* and *s*.

13. Tri-nucleotide composition (TNC)

The Tri-Nucleotide Composition gives 64 descriptors. It is defined as:

$$D(r,s,t) = \frac{N_{rst}}{N-2},\ r,s,t \in \{A,C,G,T(U)\}$$

where N_{rst} is the number of tri-nucleotide represented by nucleic acid types *r, s* and *t*.

14. zCurve mathematical formula (zCurve)

Z-curve theory is often used in genomic sequence analysis. It has got three components in three axis. They are defined as following.

$$\begin{cases} x\ axis = (\sum A + \sum G) - (\sum C + \sum T) \\ y\ axis = (\sum A + \sum C) - (\sum G + \sum T) \\ z\ axis = (\sum A + \sum T) - (\sum G + \sum C) \end{cases}$$

Three features will generate using the zCurve method.

15. Monokgap theoretical description (MonoKGap)

When -kgap=n then the $(4)\times(4)\times n$ features will exist for DNA and RNA but $(20)\times(20)\times n$ features will exist for protein. When -kgap=1, feature structure will be X_X.

When -kgap=2, feature structure will be X_X, and X__X. When -kgap=3, feature structure will be X_X. X__X, and X___X. Described with appropriate examples: When -kgap=1 then only sixteen features will exist for DNA and RNA but four hundred (400) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A, C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G, and T_T of the whole sequence of DNA respectively. When -kgap=2 then only thirty-two features will exist for DNA and RNA but eight hundred (800) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A,C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G,T_T, A__A, A__C, A__G, A__T, C__A, C__C, C__G, C__T, G__A, G__C,

G__G, G__T, T__A, T__C, T__G, and T__T of the whole sequence of DNA respectively.

When -kgap=3 then only forty-eight features will exist for DNA and RNA, but one thousand and two hundred (1,200) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A,C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G,T_T, A__A, A__C, A__G,A__T, C__A, C__C, C__G, C__T, G__A, G__C, G__G, G__T, T__A, T__C, T__G, T__T, A___A, A___C, A___G, A___T, C___A, C___C, C___G, C___T, G___A, G___C, G___G,G___T, T___A, T___C, T___G, and T___T of the whole sequence of DNA respectively.

16. MonoDiKGap Theoretical Description (MonoDiKGap)

When kgap=n then the $(4)\times(4\times4)\times n$ features will exist for DNA and RNA, but $(20)\times(20\times20)\times n$ features will exist for protein.

When -kgap=1, feature structure will be X_XX.

When -kgap=2, feature structure will be X_XX, and X__XX.

When -kgap=3, feature structure will be X_XX, X__XX, and X___XX. Described with appropriate examples:

When -kgap=1 then only sixty-four (64) features will exist for DNA and RNA, but eight thousand (8,000) features will exist for protein. Features will be numbers of A_AA, A_AC,A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA, A_GC, A_GG, A_GT, A_TA, A_TC, A_TG,A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT,C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA,G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC,T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG, and T_TT of the whole sequence of DNA respectively.

When -kgap=2 then only hundred and twenty-eight (128) features will exist for DNA and RNA, but sixteen thousand (16,000) features will exist for protein. Features will be numbers of A_AA, A_AC,A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA,

A_GC, A_GG, A_GT, A_TA, A_TC, A_TG,A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT,C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA,G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC,T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG,T_TT, A__AA, A__AC, A__AG, A__AT, A__CA, A__CC, A__CG, A__CT, A__GA, A__GC,A__GG, A__GT, A__TA, A__TC, A__TG, A__TT, C__AA, C__AC, C__AG, C__AT, C__CA,C__CC, C__CG, C__CT, C__GA, C__GC, C__GG, C__GT, C__TA, C__TC, C__TG, C__TT,G__AA, G__AC, G__AG, G__AT, G__CA, G__CC, G__CG, G__CT, G__GA, G__GC, G__GG,G__GT, G__TA, G__TC, G__TG, G__TT, T__AA, T__AC, T__AG, T__AT, T__CA, T__CC,T__CG, T__CT, T__GA, T__GC, T__GG, T__GT, T__TA, T__TC, T__TG, and T__TT of the whole sequence of DNA respectively.

RNA feature extraction method

SeqFea-Learn contains 15 RNA feature extraction methods.

Feature Extraction Method	Method Number
Kmer	1
Reverse compliment Kmer (RCKmer)	2
Nucleic acid composition (NAC)	3
Di-nucleotide composition (DNC)	4
Tri-Nucleotide Composition (TNC)	5
Binary encoding (BE)	6
zCurve Mathematical Formula (zCurve)	7
Dinucleotide based auto covariance (DAC)	8
Dinucleotide based cross covariance (DCC)	9
Di-nucleotide based auto-cross covariance (DACC)	10
Position-specific dinucleotide propensity (PSDNP)	11
Position-specific trinucleotide propensity (PSTNP)	12
MonoKGap theoretical description (MonoKGap)	13
MonoDiKGap theoretical description (MonodiKGap)	14
Pseudo di-nucleotide composition (PseDNC)	15

1. Kmer

For kmer descriptor [1, 2], the RNA sequences are represented as the occurrence frequencies of k neighboring nucleic acids. The Kmer (k=3) descriptor can be calculated as:

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

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

2. Reverse compliment kmer (RCKmer)

The reverse compliment kmer [1, 3] is a variant of kmer descriptor, in which the kmers are not
expected to be strand-specific. For instance, for a DNA sequence, there are 16 types of 2-mers (i.e. 'AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'CT', 'GA', 'GC', 'GG', 'GT', 'TA', 'TC', 'TG', 'TT'), ‘TT’ is reverse compliment with ‘AA’. After removing the reverse compliment kmers, there are only 10 distinct kmers in the reverse compliment kmer approach ('AA', 'AC', 'AG', 'AT', 'CA', 'CC', 'CG', 'GA', 'GC', 'TA').

3. Pseudo dinucleotide composition (PseDNC)

PseDNC [4] is an approach incorporating the contiguous local sequence-order information and the global sequence-order information into the feature vector of the RNA sequence. Given a RNA sequence D, the PseDNC feature vector of D is defined:

$$D=[d_1,d_2,d_3,\cdots,d_{16},d_{16+1},\cdots,d_{16+\lambda}]$$

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

where f_k is the normalized occurrence frequency of dinucleotides 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 ranged from 0 to 1, θ_j is called the correlation factor that reflects the sequence-order correlation between all the most contiguous dinucleotides along a RNA sequence, which is defined:

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

where the correlation function is given by

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

where μ is the number of physicochemical indices, in this approach, 6 indices reflecting the local RNA structural properties are employed to generate the PseDNC feature vector, $P_u(R_iR_{i+1}),P_u(R_jR_{j+1})$ represents the numerical value of the $u\text{-}th$ physicochemical index of the dinucleotide R_iR_{i+1},R_jR_{j+1}.

4. Dinucleotide-based auto covariance (DAC)

The DAC [5, 6] measures the correlation of the same physicochemical index between two dinucleotides separated by a distance of lag along the sequence, which can be calculated as:

$$DAC(u,l)=\sum_{i=1}^{L-l-1}(P_u(R_iR_{i+1})-\bar{P}_u)(P_u(R_{i+l}+R_{i+l+1})-\bar{P}_u)/(L-l-1)$$

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

In such a way, the length of DAC feature vector is $N\times lag$, where N is the number of physicochemical indices, and lag is the maximum of sequence.

5. Dinucleotide-based cross covariance (DCC)

Given a RNA sequence, the DCC approach [5, 6] measures the correlation of two different physicochemical indices between two dinucleotides separated by lag nucleic acids along the sequence, which can be calculated by:

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

Where u_1,u_2 are two different physicochemical indices, L is the length of the RNA sequence, $P_{u_1}(R_iR_{i+1}),P_{u_2}(R_iR_{i+1})$ is the numerical value of the physicochemical index u_1,u_2 for the dinucleotide R_iR_{i+1}. $\bar{P}_{u_1}$ is the average value for physicochemical index value u_1 along the whole sequence:

$$\overline{P_u}=\sum_{j=1}^{L-1}P_u(\mathrm{R}_j\mathrm{R}_{j+1})/(L-1)$$

In such a way, the length of the DCC feature vector is $N\times(N-1)\times lag$, lag is the maximum of sequence; N is the number of physicochemical indices.

6. Dinucleotide-based auto-cross covariance (DACC)

DACC is a combination of DAC and DCC [5, 6]. Therefore, the length of the DACC feature vector is $N\times N\times lag$, where N is the number of physicochemical indices and lag is

the maximum of sequence.

7. Nucleic acid composition (NAC)

The Nucleic acid composition (NAC) [7] encoding calculates the frequency of each nucleic acid type in a nucleotide sequence. The frequencies of all 4 natural nucleic acids (i.e. "ACGT or U") can be calculated as:

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

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

8. Di-nucleotide composition (DNC)

The Di-Nucleotide Composition [7] gives 16 descriptors. It is defined as:

$$D(r,s) = \frac{N_{rs}}{N-1},\ r,s \in \{A, C, G, T(U)\}$$

where N_{rs} is the number of di-nucleotide represented by nucleic acid types r and s.

9. Tri-nucleotide composition (TNC)

The Tri-Nucleotide Composition [7] gives 64 descriptors. It is defined as:

$$D(r,s,t) = \frac{N_{rst}}{N-2},\ r,s,t \in \{A, C, G, T(U)\}$$

where N_{rst} is the number of tri-nucleotide represented by nucleic acid types r, s and t.

10. zCurve mathematical formula (zCurve)

Z-curve theory [8] is often used in genomic sequence analysis. It has got three components in three axis. They are de_ned as following.

$$\begin{cases} x\ axis = (\sum A + \sum G) - (\sum C + \sum T) \\ y\ axis = (\sum A + \sum C) - (\sum G + \sum T) \\ z\ axis = (\sum A + \sum T) - (\sum G + \sum C) \end{cases}$$

Three features will generate using the zCurve method.

11. MonoKGap Theoretical Description (MonoKGap)

When kgap=n then the $(4)\times(4)\times n$ features will exist for RNA [8].

When -kgap=1, feature structure will be X_X.

When -kgap=2, feature structure will be X_X, and X__X.

When -kgap=3, feature structure will be X_X. X__X, and X___X.

Described with appropriate examples:

When -kgap=1 then only sixteen (16) features will exist for DNA and RNA but four hundred

(400) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A,
C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G, and T_T of the whole sequence of
DNA respectively.

When -kgap=2 then only thirty two (32) features will exist for DNA and RNA but eight
hundred (800) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A,C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G,T_T, A__A, A__C, A__G,
A__T, C__A, C__C, C__G, C__T, G__A, G__C, G__G, G__T, T__A, T__C, T__G, and T__T of
the whole sequence of DNA respectively.

When -kgap=3 then only forty eight (48) features will exist for DNA and RNA, but one
thousand and two hundred (1,200) features will exist for protein. Features will be numbers of A_A, A_C, A_G, A_T, C_A,C_C, C_G, C_T, G_A, G_C, G_G, G_T, T_A, T_C, T_G,T_T, A__A, A__C, A__G,A__T, C__A, C__C, C__G, C__T, G__A, G__C, G__G, G__T, T__A, T__C, T__G, T__T, A___A, A___C, A___G, A___T, C___A, C___C, C___G, C___T, G___A, G___C, G___G,G___T, T___A, T___C, T___G, and T___T of the whole sequence of DNA respectively.

12. MonoDiKGap Theoretical Description (MonoDiKGap)

When -kgap=n then the $(4)\times(4\times4)\times n$ features will exist for RNA [8].

When -kgap=1, feature structure will be X_XX.

When -kgap=2, feature structure will be X_XX, and X__XX.

When -kgap=3, feature structure will be X_XX, X__XX, and X___XX.

Described with appropriate examples:

When -kgap=1 then only sixty four (64) features will exist for DNA and RNA, but eight thousand (8,000) features will exist for protein. Features will be numbers of A_AA, A_AC,A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA, A_GC, A_GG, A_GT, A_TA, A_TC, A_TG,A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT,C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA,G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC,T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG, and T_TT of the whole sequence of DNA respectively.

When -kgap=2 then only hundred and twenty eight (128) features will exist for DNA and

RNA, but sixteen thousand (16,000) features will exist for protein. Features will be numbers of A_AA, A_AC,A_AG, A_AT, A_CA, A_CC, A_CG, A_CT, A_GA, A_GC, A_GG, A_GT, A_TA, A_TC, A_TG,A_TT, C_AA, C_AC, C_AG, C_AT, C_CA, C_CC, C_CG, C_CT, C_GA, C_GC, C_GG, C_GT,C_TA, C_TC, C_TG, C_TT, G_AA, G_AC, G_AG, G_AT, G_CA, G_CC, G_CG, G_CT, G_GA,G_GC, G_GG, G_GT, G_TA, G_TC, G_TG, G_TT, T_AA, T_AC, T_AG, T_AT, T_CA, T_CC,T_CG, T_CT, T_GA, T_GC, T_GG, T_GT, T_TA, T_TC, T_TG,T_TT, A__AA, A__AC, A__AG, A__AT, A__CA, A__CC, A__CG, A__CT, A__GA, A__GC,A__GG, A__GT, A__TA, A__TC, A__TG, A__TT, C__AA, C__AC, C__AG, C__AT, C__CA,C__CC, C__CG, C__CT, C__GA, C__GC, C__GG, C__GT, C__TA, C__TC, C__TG, C__TT,G__AA, G__AC, G__AG, G__AT, G__CA, G__CC, G__CG, G__CT, G__GA, G__GC, G__GG,G__GT, G__TA, G__TC, G__TG, G__TT, T__AA, T__AC, T__AG, T__AT, T__CA, T__CC,T__CG, T__CT, T__GA, T__GC, T__GG, T__GT, T__TA, T__TC, T__TG, and T__TT of the whole sequence of DNA respectively.

Protein feature extraction method

SeqFea-Learn contains 31 Protein feature extraction methods.

Protein feature extraction methods	Method Number
Amino acid composition (AAC)	1
Dipeptide composition (DC)	2
Composition of k-spaced amino acid pairs (CKSAAP)	3
Grouped dipeptide composition (GDC)	4
Grouped tripeptide composition (GTC)	5
Conjoint triad (CT)	6
K-spaced conjoint triad (KSCTriad)	7
Composition (C)	8
Transition (T)	9
Distribution (D)	10
Encoding based on grouped weight (EBGW)	11
Auto covariance (AC)	12
Moreau-Broto autocorrelation (Morean-Broto)	13
Moran autocorrelation (Moran)	14
Geary autocorrelation (Geary)	15
Quasi-sequence-order (QSO)	16
Pseudo-amino acid composition (PseAAC)	17
Amphiphilic pseudo-amino acid composition (APAAC)	18
Amino acid composition PSSM (AAC-PSSM)	19
Dipeptide composition PSSM (DPC-PSSM)	20

Bi-gram PSSM (Bi-PSSM)	21
Auto covariance PSSM (AC-PSSM)	22
Pseudo PSSM (PsePSSM)	23
AB-PSSM	24
Secondary structure composition (SSC)	25
Accessible surface area composition (ASA)	26
Torsional angles composition (TAC)	27
Torsional angles bigram (TA-bigram)	28
Structural probabilities bigram (SP-bigram)	29
Torsional angles auto-covariance (TAAC)	30
Structural probabilities auto-covariance (SPAC)	31

Type 1: Amino acid composition information-based methods

1. Amino acid composition (AAC)

The amino acid composition is the fraction of each amino acid type within a protein. The amino acid composition gives 20 features and the fractions of all 20 natural amino acids are calculated as

$$f(r) = \frac{N_r}{N}, \quad r = 1, 2, 3, \ldots, 20$$

where N_r is the number of the amino acid type r and N is the length of the sequence.

2. Dipeptide composition (DC)

The dipeptide composition is used to transform the variable length of proteins to fixed length feature vectors.A dipeptide composition has been used earlier by Grassmann et al.

and Reczko and Bohr for the development of fold recognition methods.We adopt the same dipeptide composition-based approach in developing a deep neural networks-based method for predicting protein-protein inter-action.The dipeptide composition gives a fixed pattern length of 400.Dipeptide composition encapsulates information about the fraction of amino acids as well as their local order.The dipeptide composition is defined as

$$fr(r,s) = \frac{N_{(r,s)}}{N}, \quad r,s = 1,2,3,\ldots,20$$

where N the number of dipeptide represented by amino acid type r and s .

3. g -gap dipeptide composition introduces (g-GapDC)

The g -gap dipeptide composition introduces (g -Gap DC) extracts important intrinsic correlation information of protein sequences in multidimensional space. For a protein P , L of the sequence length C, the calculation process of the g -Gap DC can be expressed as

$$f_{\varepsilon}^{g} = \frac{n_{\varepsilon}^{g}}{\sum_{\varepsilon=1}^{400} n_{\varepsilon}^{g}} = \frac{n_{\varepsilon}^{g}}{L-g-1}$$

$$\mathrm{P} = \{ f_1^g, f_2^g, \cdots, f_{\varepsilon}^g, \cdots, f_{400}^g \}^{\mathrm{T}}$$

where g represents the number of residues in the primary structure of the two amino acids, n_{ε}^{g} represents the number of occurrences of the $\varepsilon(\varepsilon = 1,2,\cdots,400)$ -th feature in the g -gap dipeptide, and f_{ε}^{g} represents the frequency at which the ε -th feature in the g -gap dipeptide appears in the sequences. When $g = 0$, there is no gap between two adjacent amino acid residues, and when $g = 1$, it means that one amino acid residue is between two

adjacent amino acid residues. From equation (9), a protein sequence can be represented as a 400-dimensional feature vector.

4. Grouped di-peptide composition (GDC)

The Grouped di-peptide composition encoding is another variation of the DPC descriptor. It is composed of a total of 25 descriptors that are defined as:

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

where N_{rs} is the number of the di-peptide type r,s, and N is the length of the sequence.

5. Grouped tri-peptide composition (GTC)

The Grouped tri-peptide composition encoding is also a variation of TPC descriptor, which generates 125 descriptors, defined as:

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

where N_{rst} is the number of the tri-peptide type r,s,t, and N is the length of the sequence.

6. Conjoint triad (CT)

First, 20 amino acids are clustered into seven classes based on dipoles and volumes of side chains. Considering the interaction between the amino acid and its vicinal amino acids, the three continuous amino acids are regarded as a unit, so that we can obtain $7\times7\times7=343$ triad types. We calculate the frequency of occurrence of each triad called $f_i\,(i=1,2,\cdots,343)$. Then the 343-dimensional feature vector is obtained according to equation (6).

$$d_i = \frac{f_i - \min\{f_1, f_2, \cdots, f_{343}\}}{\max\{f_1, f_2, \cdots, f_{343}\}}, i = 1, 2, \cdots, 343$$

7. *k*-Spaced Conjoint Triad (KSCTriad)

The *k*-spaced conjoint triad descriptor is based on the conjoint ctriad descriptor,

which not only calculates the numbers of three continuous amino acid units, but also considers the continuous amino acid units that are separated by any *k* residues (The default maximum value of *k* is set to 5). For example, AxRxT is a 1-spaced triad. Thus, the dimensionality of the KSCTriad encoded feature vector is 343 (*k*+1).

8. Composition, transition and distribution (CTD)

8.1 Composition

For CTD, taking hydrophobicity as an example, all amino acids are classified into three categories: polar, neutral, and hydrophobic. The replacement sequence consists of three types, and the composition descriptors of the polar, neutral, and hydrophobic residues of the protein can be calculated as follows:

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

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

8.2 Transition

The transition descriptor first converts the original sequence into a replacement sequence, and T includes three characteristics, the dipeptide composition frequency from the polar group to the neutral group and the composition frequency from the neutral group to the polar group. Transitions between the neutral group and the hydrophobicity and these

between hydrophobic group and the polar group are defined in the same way. The T descriptor is defined as follows:

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

Where $N(r,s)$ and $N(s,r)$ are the rs and sr dipeptide frequency, respectively. N is the sequence length.

8.3 Distribution

For each group (polar, neutral and hydrophobic), the D descriptor can generate five values. We obtain the position of the first, 25%, 50%, 75% and 100% of the specific encoded group sequence and then divided the position by the whole sequence. Given sequence MTTTVPKVFAFHEF. It can be represented as '32223213323213' according to Hydrophobicity_PRAM900101. '1' represents polar, '2' represents neutral, '3' represents hydrophobicity. Take '3' for example, there are 6 residues encoded '3'. The first '3' is 1. The second '3' is $25\%\times 6=1$. The third '3' is $50\%\times 6=3$. The fourth '3' is $75\%\times 6=4$. The fifth '3 is $100\%\times 6=6$. The position in the first, the second, the third, the fourth, the fifth '3' of whole sequence are 1, 1, 8, 9, 14, respectively. So the distribution descriptor for '3' are $(1/14),(1/14),(8/14),(9/14),(14/14)$.

The C descriptor generates a 39-dimensional feature vector, the T descriptor generates a 39-dimensional feature vector, and the D descriptor generates a 195-dimensional feature vector. For each protein sequence, the CTD generates a 273-dimensional feature vector.

9. Encoding based on grouped weight coding (EBGW)

Studies have shown that different amino acids have different physical and chemical properties. The EBGW can capture the sequence and physicochemical information based on grouped situation. These amino acids are classified into four categories:

neutral and hydrophobic amino acids $C1=\{G, A,V,L,I ,M,P,F,W\}$

neutral and polarity amino acids $C2=\{Q,N,S,T,Y,C\}$

acidic amino acids $C3=\{D,E\}$

basic amino acids $C4=\{H,K,R\}$.

Thus, we can get three combinations, each of which can partition the 20 amino acid residues into two disjoint group: $\{C1,C2\}$ vs $\{C3,C4\}$, or $\{C1,C3\}$ vs $\{C2,C4\}$, and $\{C1,C4\}$ vs $\{C2,C3\}$. Let $P: R_1R_2\cdots R_L$ be a protein sequence, we can transform it into three binary sequences by three homomorphic maps $\Phi_i(P)=\Phi_i(R_1),\Phi_i(R_2),...,\Phi_i(R_L)(i=1,2,3)$ which are defined as follows:

$$\Phi_1(R_j)=\begin{cases}1 & if\ \ R_j\in C1+C2\\ 0 & if\ \ R_j\in C3+C4\end{cases}\ (j=1,2,...,L)$$

$$\Phi_2(R_j)=\begin{cases}1 & if\ \ R_j\in C1+C3\\ 0 & if\ \ R_j\in C2+C4\end{cases}\ (j=1,2,...,L)$$

$$\Phi_3(R_j)=\begin{cases}1 & if\ \ R_j\in C1+C4\\ 0 & if\ \ R_j\in C2+C3\end{cases}\ (j=1,2,...,L)$$

where $H_i(P)=\Phi_i(P)=h_1^i,h_2^i,...,h_L^i(i=1,2,3)$, and we call $H_i(P)(i=1,2,3)$ as characteristic sequence of the protein sequence.

Then $H_1(P)$, $H_2(P)$, $H_3(P)$ are three binary sequences of length L. These sequences are divided into a number of sub-sequences of increasing length successively. A fixed parameter N is set and the sub-sequence can be expressed as $\lfloor \underline{kL/N} \rfloor (k=1,2,..,N)$, where

$\lfloor . \rfloor$ represents the integer operator. Calculate the frequency of 1 in each sub-sequence, each $H_i(P)$ can be converted into an *N*-dimensional feature vector. To sum up, for a protein sequence *P* with length *L*, a 3*N*-dimension vector can be obtained.

Type 2: Physicochemical information-based methods

10. Auto covariance (AC)

Suppose a protein sequence P with *L* amino acid residues; i.e.

$$P = R_1 R_2 R_3 R_4 \cdots R_L$$

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 AC approach measures the correlation of the same property between two residues separated by a distance of l along the sequence, which can be calculated as:

$$AC(i,l) = \sum_{i=1}^{L-l} (P_u(R_i) - \bar{P}_u)(P_u(R_{i+l}) - \bar{P}_u) / (L-l)$$

where u is a physicochemical index, L is the length of the protein sequence, means the numerical value of the physicochemical index u for the amino acid R_i at position i, $\bar{P}_u$ is the average value for physicochemical index u along the whole sequence. In such a way, the length of AC feature vector is $N \times lag$, where *N* is the number of physicochemical indices extracted from AAindex; lag is the maximum of sequence.

11. Moreau-Broto autocorrelation (Morean-Broto)

Moreau-Broto autocorrelation descriptor is defined as:

$$NMBA(l) = \frac{NBA(l)}{N-l},\ l = 1, 2, \mathrm{L}\ , lag$$

where $MBA(l) = \sum_{i=1}^{N-l} P(AA_i) P(AA_{i+l})$, AA_i and AA_{i+l} indicate the *i-th* and the $i+l$-*th* amino acids of the protein sequence, respectively. $P(AA_i)$ and $P(AA_{i+l})$ indicate the normalized

physicochemical values of AA_i and AA_{i+l}. The lag is the parameter that needs to be adjusted.

12. Moran autocorrelation (Moran)

Moran autocorrelation descriptor is defined as:

$$MA(l) = \frac{\frac{1}{N-l}\sum_{i=1}^{N-l}(P(AA_i) - \bar{P})(P(AA_{i+l}) - \bar{P})}{\frac{1}{N}\sum_{i=1}^{N}(P(AA_i) - \bar{P})^2}, l = 1, 2, \mathrm{L}\ , lag$$

where $\bar{P}$ represents the mean value of whole protein sequence for specific physicochemical property.

13. Geary autocorrelation (Geary)

Geary autocorrelation descriptor is defined as:

$$GA(l) = \frac{\frac{1}{2(N-l)}\sum_{i=1}^{N-l}(P(AA_i) - P(AA_{i+l}))^2}{\frac{1}{N}\sum_{i=1}^{N}(P(AA_i) - \bar{P})^2}, l = 1, 2, \mathrm{L}\ , lag$$

14. Quasi-sequence-order descriptors (QSO)

The sequence order features can also be used for representing amino acid distribution patterns of a specific physicochemical property along protein or peptide sequence. These descriptors are derived from both the Schneider-Wrede physicochemical distance matrix and the Grantham chemical distance matrix between each pair of the 20 amino acids. The d-th rank sequence-order-coupling number is defined as

$$\tau_d = \sum_{i=1}^{N-d}\left(d_{i,i+d}\right)^2, \quad d = 1, 2, 3, \ldots, \max lag$$

where $d_{i,i+d}$ is the distance between the two amino acids at position i and $i+d$. Maxlag is the maximum lag and the length of the protein must be not less than maxlag. The maxlag is equal to 30 in the experiment. For each amino acid type, the type-1 quasi-sequence-order descriptor can be defined as

$$X_r = \frac{f_r}{\left(\sum_{r=1}^{20} f_r + w\sum_{d=1}^{\max lag} \tau_d\right)}, r = 1,2,3,\ldots,20$$

where is the normalized occurrence of amino acid type-1 and is a weighting factor $(w = 0.1)$. The type-2 quasi-sequence-order is defined as

$$X_r = \frac{w\tau_{d-20}}{\left(\sum_{r=1}^{20} f_r + w\sum_{d=1}^{\max lag} \tau_d\right)}, r = 21,22,23,\ldots,20+\max lag$$

In addition to the Schneider-Wrede physicochemical distance matrix used by Chou et al., another chemical distance matrix by Grantham is also used here. The sequence-order features produce a total of $(30+50)\times 2=160$ descriptors.

15. Pseudo-amino acid composition (PseAAC)

Pseudo-amino acid composition are utilized to extract the physicochemical information. Pseudo-amino acid composition (PseAAC) represents the composition information of the protein and sequence order information. The feature vector of PseAAC can be represented:

$$X = [x_1, x_2, \mathrm{L}\ , x_{19}, x_{20}, x_{20+1}, x_{20+\lambda}]^T\ (\lambda < L)$$

where the first 20-dimentional feature vector represents the amino acid composition information, and the latter λ dimensional vector represents the order information of protein sequence. L is the length of amino acid sequence.

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

where f_i is the normalized frequency of 20 amino acids in protein. θ_j is the layer sequence correlation factor calculated according to the equation (2). Sequence correlation factor can be obtained from the hydrophobicity, hydrophilicity, and side-chain mass of the amino acid.

16. Amphiphilic pseudo amino acid composition (APAAC)

APAAC (http://www. csbio. sjtu. edu. cn/bioinf/PseAAC/type2. htm) are also called type-2 pseudoamino acid composition. The definitions of these qualities are similar to PAAC descriptors. First, two variables are derived from the original hydrophobicity values $H_1^0(i)$ and hydrophilicity value $H_2^0(i)$ of 20 amino acids $(i = 1, 2, \ldots, 20)$:

$$H_1(i) = \frac{H_1^0(i) - \sum_{i=1}^{20} \frac{H_1^0(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20} \left[H_1^0(i) - \sum_{i=1}^{20} \frac{H_1^0(i)}{20} \right]^2}{20}}}$$

$$H_1(i) = \frac{H_2^0(i) - \sum_{i=1}^{20} \frac{H_2^0(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20} \left[H_2^0(i) - \sum_{i=1}^{20} \frac{H_2^0(i)}{20} \right]^2}{20}}}$$

where $H_1(i)$ and $H_2(i)$,the hydrophobicity and hydrophilicity correlation functions, are defined respectively as

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

where sequence order factors can be defined as

$$\tau_1 = \frac{1}{N-1}\sum_{i=1}^{N-1} H_{i,i+1}^1, \quad \tau_2 = \frac{1}{N-1}\sum_{i=1}^{N-2} H_{i,i+1}^2$$

$$\tau_3 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^1, \qquad \tau_4 = \frac{1}{N-2}\sum_{i=1}^{N-2} H_{i,i+2}^2$$

$$\tau_{2\lambda-1} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^1, \quad \tau_{2\lambda} = \frac{1}{N-\lambda}\sum_{i=1}^{N-\lambda} H_{i,i+\lambda}^2 (\lambda < N)$$

Then a set of descriptors called "Amphiphilic Pseudo Amino Acid Composition" are defined as

$$p^u = \frac{f_u}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda} \tau_j} (1 < u < 20)$$

$$p^u = \frac{w\tau_u}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda} \tau_j} (20+1 < u < 20+2\lambda)$$

where w is the weight factor and is taken as w=0.5, and λ is equal to 30 in the experiment. So, we produce a total of $20+2\times30=80$ descriptors.

Type 3: Evolutionary information-based method

In order to obtain the evolutionary information of the amino acid sequence, all the protein sequences in the dataset are compared with the non-redundant database SwissProt using the PSI-BLAST program. The program can search sequences based on the iterative BLAST search method. Evolutionary information in the position specific scoring matrix (PSSM) plays an important role in biological system analysis.

During the running process, the parameter E-value threshold of PSI-BLAST is set to 0.001, the maximum number of iterations is set to 3, and the remaining parameters are set

by default. Then PSSM of each protein sequence is obtained. For a protein sequence whose length is L, PSSM is shown in equation (49).

$$P_{PSSM} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{L,1} & p_{L,2} & \cdots & p_{L,20} \end{bmatrix}$$

where each row of the PSSM represents a log likelihood score for amino acid substitutions occurring at corresponding positions in the query sequence. Where $P_{i,j}$ represents the i-th position of query sequence being mutated to type j during evolution process. The scores are positive integers or negative integers. A positive integer indicates that more mutations have occurred in the alignment and a negative integer indicates that fewer substitutions have occurred in the alignment.

17. Amino acid composition PSSM (AAC-PSSM)

In this calculating process of amino acid composition PSSM (AAC-PSSM), the PSSM is standardized by the logic function. PSSM elements are map to the interval [0,1].

$$f(x) = \frac{1}{1+e^{-x}}$$

PSSM are converted to feature vector by AAC-PSSM via equation (51)

$$P_{AAC} = (p_1, p_2, \cdots, p_j, \cdots p_{20})^T \quad (j = 1, 2, \cdots, 20)$$

where $P_j = \frac{1}{L}\sum_{i=1}^{L} p_{ij}$ $(j = 1, 2, \cdots 20)$, p_j represents the composition information of the j amino acid residue, which is the average score of j-th amino acid in PSSM.

18. Dipeptide composition PSSM (DPC-PSSM)

The PSSM contains evolutionary information. ACC-PSSM only represents the composition information from PSSM, and loses the order information, which is insufficient to fully represent the evolutionary information. Dipeptide composition PSSM (DPC-PSSM) can reflect the sequence-order information in the PSSM, which converts the character signal into the numerical signal, and the extracted feature vector can be expressed as

$$P_{DPC} = (D_{1,1}, D_{1,2}, \cdots, D_{1,20}, D_{2,1}, D_{2,2}, \cdots, D_{2,20}, \cdots, D_{20,20})$$

where $D_{i,j} = \frac{1}{L-1}\sum_{k=1}^{L-1} p_{k,i} \times p_{k+1,l}$, the dimension of DPC-PSSM is 400.

19. Bi-gram PSSM (Bi-PSSM)

For Bi-gram PSSM, the frequency of the transition from the m amino acids to the n amino acids is calculated:

$$B_{m,n} = \sum_{i=1}^{L-1} p_{i,m} p_{(i+1),n}, m, n = 1, 2, \cdots, 20$$

Therefore, there are 400 possible cases for $B_{m,n}$, then the Bi-gram PSSM eigenvector for each protein sequence is:

$$F = [B_{1,1}, B_{1,2}, \cdots, B_{1,20}, B_{2,1}, \cdots, B_{2,20}, \cdots, B_{20,1}, \cdots, B_{20,20}]^T$$

20. Auto covariance PSSM (AC-PSSM)

AC-PSSM can transform the PSSMs of different lengths into fixed-length vector. The AC variable measures the correlation of the same property between two residues separated by a distance of lag along the sequence, which can be calculated as:

$$AC(i,l) = \sum_{j=1}^{L-l} \left(S_{i,j} - \bar{S}_i\right)\left(S_{i,j+l} - \bar{S}_i\right) / (L-l)$$

where i is one of the residues, L is the length of the protein sequence, $S_{i,j}$ is the PSSM score of amino acid i at position j, $\overline{S_i}$ is the average score for amino acid i along the whole sequence:

$$\overline{S_i} = \sum_{j=1}^{L} S_{i,j} / L$$

In such a way, the number of AC variables can be calculated as $20 \times lag$, where lag is the maximum of sequence.

21. Cross covariance PSSM (CC-PSSM)

CC-PSSM can transform the PSSMs of different lengths into fixed-length vectors. The CC variable measures the correlation of two different properties between two residues separated by lag along the sequence, which can be calculated by:

$$CC(i1, i2, l) = \sum_{j=1}^{L-l} \left(S_{i1,j} - \overline{S}_{i1}\right)\left(S_{i2,j+l} - \overline{S}_{i2}\right) / (L - l)$$

where $i1, i2$ are two different amino acids and $\overline{S}_{i1}\left(\overline{S}_{i2}\right)$ is the average score for amino acid $i1(i2)$ along the sequence. Since the CC variables are not symmetric, the total number of CC variables is $380 \times lag$.

22. Auto-cross covariance PSSM (ACC-PSSM)

ACC-PSSM as one of the multivariate modeling tools, can transform the PSSMs of different lengths into fixed-length vectors by measuring the correlation between any two properties. ACC results in two kinds of variables: AC between the same property, and cross-covariance (CC) between two different properties. Each protein sequence is

represented as a vector of either AC variable or ACC variable that is a combination of AC and CC.

23. Pseudo PSSM (PsePSSM)

According to the pseudo amino acid composition, we obtain the PsePSSM feature vector:

$$\overline{p}_j = \begin{cases} \frac{1}{L}\sum_{i=1}^{L} p_{i,j} & j = 1,2,\mathrm{L}\ ,20,\ \zeta = 0 \\ \frac{1}{L-\zeta}\sum_{i=1}^{L-\zeta}(p_{i,j} - p_{i+\zeta,j})^2 & j = 1,2,\mathrm{L}\ ,20,\ \zeta < L \end{cases}$$

where, each protein sequence can generate $20+20\times\xi$ the dimensional feature vector. The first 20-dimensional vector represents the composition information of the PSSM matrix, and the remaining $20\times\xi$ dimensional feature vector represents the order evolutionary information. PsePSSM can transform an inconsistent protein sequence into a consistent numerical vector by feature extraction.

24. AB-PSSM

AB-PSSM is based on the averaged PSSM profiles over blocks, each with 5 percent of a sequence. Thus, a protein sequence, regardless of its length, is divided into 20 blocks and each block consists of 20 features (derived from the 20 columns in PSSMs). Mathematically, for the *j-th* block, the feature F_j is a 1×20 dimensional feature vector, which is generated by using the following equation:

$$F_j = \frac{1}{B_j}\sum_{i=1}^{B_j} P_i^j$$

where B_j is the size of the *j-th* block, which is 5 percent of the length of a sequence and Pej i is a 1×20 vector extracted from the PSSM profile at the *i-th* position in the *j-th*

block. For each sequence, there are a total of 20 blocks; therefore, the final feature is a 400-dimensional vector.

Type 4: Structural information-based methods

25. Secondary structure composition (SSC)

This feature is the normalized count or frequency of the structural motifs present at the amino-acid residue positions. There are three types of motifs: α-helix (H), β-sheet (E) and random coil (C). SPIDER2 returns a vector *SS* of dimension $L \times 1$ containing this information. Thus, we can define this feature as following:

$$SS - Composition(i) = \frac{1}{L}\sum_{j=1}^{L} c_{ij}, 1 \le i \le 3$$

where, L is the length of the protein and

$$c_{ij} \begin{cases} 1, & if \quad SS_j = f_i \\ 0, & else \end{cases}$$

where, *SSj* is the structural motif at position *j* of the protein sequence and *fi* is one of the 3 different motif symbols.

26. Accessible surface area composition (ASA)

The accessible surface area composition is the normalized sum of accessible surface area defined by:

$$ASA - Composition = \frac{1}{L}\sum_{i=1}^{L} ASA(i)$$

where ASA is the vector of accessible surface area of dimension $L \times 1$ containing the values of accessible surface area for all the amino acid residues.

27. Torsional angles composition (TAC)

Four different types of torsional angles: ϕ, ψ, τ and θ are returned by SPIDER2 for each residue. First, we convert each of them into radians from degree angles and then take sign and cosine of the angles at each residue position. Thus, we get a matrix of dimension. We denotethis matrix by *T*. Torsional angles composition is defined as

$$TA-Composition(k)=\frac{1}{L}\sum_{i=1}^{L}T_{i,k}(1\le k\le 8)$$

28. Torsional angles bigram (TA-bigram)

The Bigram for the torsional angles is similar to that of the PSSM matrix and is defined as:

$$TA-bigram(k,l)=\frac{1}{L}\sum_{i=1}^{L-1}T_{i,k}T_{i+1,l}(1\le k\le 8,1\le l\le 8)$$

29. Structural probabilities bigram (SP-bigram)

Structural probabilities for each position of the amino-acid residue are given in the SPD2 file as a matrix of dimension $L \times 3$, which we denote by *P*. The Bigram of the structural probabilities is similar to that of PSSM matrix and is defined as:

$$SP-bigram(k,l)=\frac{1}{L}\sum_{i=1}^{L-1}P_{i,k}P_{i+1,l}(1\le k\le 3,1\le l\le 3)$$

30. Torsional angles auto-covariance (TAAC)

This feature is also derived from the torsional angles and is defined as:

$$TA-Auto-Co\,\mathrm{var}\,iance(k,j)=\frac{1}{L}\sum_{i=1}^{L-k}T_{i,j}T_{i+k,j}(1\le j\le 8,1\le k\le DF)$$

31. Structural probabilities auto-covariance (SPAC)

This feature is also derived from the structural probabilities and is defined as:

$$SP-Auto-Co\,\mathrm{var}\,iance(k,j)=\frac{1}{L}\sum_{i=1}^{L-k}P_{i,j}P_{i+k,j}(1\le j\le 3,1\le k\le DF)$$

Feature selection methods	Method Number
LASSO	1
Elastic net (EN)	2
L1-SVM	3
L1-LR	4
Extra-Trees-RFE	5
XGBosst-RFE	6
SVM-RFE	7
LR-RFE	8
Mutual information (MI)	9
Minimum redundancy maximum relevance (MRMR)	10
Joint mutual information (JMI)	11
Maximum relevance maximum distance (MRMD)	12
Information gain (IG)	13
Chi-square test (CHI2)	14
Pearson correlation (Pearson)	15
ReliefF	16
Trace Ratio	17
Gini Index	18
Spectral Feature Selection (SPEC)	19
Fisher Score	20
T-Score	21

1. Mutual Information Feature Selection (MIFS)

Mutual Information Feature Selection considers both the feature relevance and feature redundancy in the feature selection phase, the feature score for a new unselected feature X_k can be formulated as follows:

$$J_{MIFS}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in S} I(X_k;X_j)$$

In MIFS, the feature relevance is evaluated by $I(X_k;Y)$, while the second term penalizes features that have a high mutual information with the currently selected features such that feature redundancy is minimized.

2. Information gain (IG)

Information Gain measures the importance of a feature by its correlation with class labels. It assumes that when a feature has a strong correlation with the class label, it can help achieve good classification performance.

The Mutual information score for feature X_k is:

$$J_{MIM}(X_k) = I(X_k;Y)$$

It can be observed that in MIM, the scores of features are assessed individually. Therefore, only the feature correlation is considered while the feature redundancy is completely ignored. After it obtains the MIM feature scores for all features, we choose the features with the highest feature scores and add them to the selected feature set. The process repeats until the desired number of selected features is obtained.It can also be observed that MIM is a special case of linear combination of Shannon information terms in Eq. (13) where both β and λ are equal to zero.

3. Minimum redundancy maximum relevance (MRMR)

Minimum redundancy maximum relevance criterion is set to the value of β to be the reverse of the number of selected features:

$$J_{MRMR}(X_k) = I(X_k;Y) - \frac{1}{|S|} \sum_{X_j \in S} I(X_k;X_j)$$

Hence, with more selected features, the effect of feature redundancy is gradually reduced.The intuition is that with more non-redundant features selected, it becomes more difficult for new features to be redundant to the features that have already been in S. In [Brown et al. 2012], it gives another interpretation that the pairwise independence between features becomes stronger as more features are added to S, possibly because of

noise information in the data. MRMR is also strongly linked to the Conditional likelihoodmaximization framework if we iteratively revise the value of β to be $\frac{1}{|S|}$, and set the other parameter λ to be zero.

4. Joint mutual information (JMI)

MIFS and MRMR reduce feature redundancy in the feature selection process. An alternative criterion, Joint Mutual Information[Yang and Moody 1999; Meyer et al. 2008] is proposed to increase the complementary information that is shared between unselected features and selected features given the class labels. The feature selection criterion is listed as follows:

$$J_{JMI}(X_k) = \sum_{X_j \in S} I(X_k, X_j; Y)$$

The basic idea of JMI is that we should include new features that are complementary to the existing features given the class labels.JMI cannot be directly reduced to the condition likelihood maximization framework.In [Brown et al. 2012], the authors demonstrate that with simple manipulations, the JMI criterion can be re-written as:

$$J_{JMI}(X_k) = I(X_k;Y) - \frac{1}{|S|}\sum_{X_j \in S} I(X_j; X_k) + \frac{1}{|S|}\sum_{X_j \in S} I(X_j; X_k \mid Y)$$

Therefore, it is also a special case of the linear combination of Shannon information terms by iteratively setting β and λ to be $\frac{1}{|S|}$.

5. Elastic net

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$, consider the linear regression model with squared error as the loss function, and the optimization goal is:

$$\min_{\omega} \sum_{i=1}^{m} \left(y_i - \omega^T x_i\right)^2$$

The shortcoming of above loss function is that when there are many features, it is easy to tend to be overfitting, so the equation can be seen as a convex linear combination of L_1 norm and L_2 norm.

$$\min_{w} \frac{1}{2\times n}\|y - Xw\|_2^2 - \alpha \times \beta \|w\|_1 + \frac{1}{2}\alpha \times (1-\beta)\|w\|_2^2$$

where n is the number of samples. w is the weight coefficient. α, β aer the penalty parameters. Equation is called elastic net (EN). EN has good performance when dealing with the correlation between feature factors.

6. L1-regularization logistic regression

The L1-regularized logistic regression (L1-RLR) is an embedded feature selection that combines feature selection with the learner training process. It selects the optimal feature subset during the training process of the learning system. The L1 norm can be solved sparsely, which reduces the computational complexity and the risk of overfitting.

Given the sample data set $D=\{(x_1,y_1),(x_2,y_2),\mathrm{L},(x_m,y_m)\}$, x_i represent the sample feature vector, y_i represent the sample class label. Thus, the L1 logistic regression can be transformed into an unconstrained optimization problem.

$$\min_{w} \quad f(\omega)=\|\omega\|_1+C(\sum_{i=1}^{l}\log(1+e^{-\omega^T x_i})+\sum_{i:y_i=-1}\omega^T x_i)$$

where $\|\cdot\|_1$ represents L1 norm. ω represents the weight coefficient. C represents the penalty term which determines the number of selected features. The larger penalty term is, the more features are selected.

7. XGBoost feature selection

XGBoost [42] is an efficient and fast gradient boosting decision tree algorithm, which uses weighted quantile sketch, regularized learning, and cache-aware block structure tree learning for ensemble learning. For the given dataset $D=\left\{(x_i,y_i)\left(|D|=n,x_i\in R^m,y_i\in\{-1,1\}\right)\right\}$, the numbers of samples and features are n and m. The objective function can be described as:

$$L(\theta)=\sum_{i}^{n}l(y_i,\hat{y}_i)+\sum_{t=1}^{T}\Omega(f_t)$$

where L is loss function, f_t is the t-th tree, $\Omega(f_t)$ represents regularized term. The second-order Taylor term is employed to represent the loss function for t-th iteration, and the approximation can be used to optimize the loss function, as the equation (13) show:

$$L^{(t)}\ \square\ \sum_{i=1}^{k}\left[l\left(y_i,y_i^{(t-1)}+g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x)\right)\right]+\Omega(f_t)$$

where $g_i=\partial_{y^{(t-1)}}l(y_i,y^{(t-1)})$ represents the first order gradient, $h_i=\partial^2_{y^{(t-1)}}l(y_i,y^{(t-1)})$ is the second order gradient. We can see the loss function only depends on the first and second

gradient. In the training process of XGBoost, we use gain to determine the optimal split node.

$$gain = \frac{1}{2}\left[\frac{(\sum_{\in I_L} g_i)^2}{\sum_{\in I_L} h_i + \lambda} + \frac{(\sum_{\in I_R} g_i)^2}{\sum_{\in I_R} h_i + \lambda} - \frac{(\sum_{\in I} g_i)^2}{\sum_{\in I} h_i + \lambda}\right] - \gamma$$

where I_L and I_R represent the number of samples of the left and right nodes after the split node, respectively. $I = I_L \cup I_R$. λ, γ are the penalty parameters. Gain represents the gain score for each split of a tree, and the final feature importance score is calculated by the average gain. The average gain is the total gain of all trees divided by the total number of splits for each feature. The higher the feature importance score of XGBoost is, the more important the corresponding feature is.

8 ReliefF algorithm

Relief algorithm (Kira and Rendell, 1992) is currently the most efficient filter feature evaluation algorithm. It fully considers the correlation between categories and features and solves the problem of multi-class feature dimension reduction. Its main idea is to calculate the correlation between features and categories, and then update the feature weights according to the degree of correlation, and quickly process high-dimensional data to remove unnecessary features effectively. Among them, the higher the weight value, the stronger the classification ability of the feature.

When dealing with multi-class $C(c_1, c_2, c_3)$ feature selection, the ReliefF algorithm first selects d samples with the nearest distance m_i from each category, then the d samples of the same kind as m_i are combined into a set X. According to the class to which the classification belongs, the samples of different classes of m_i respectively constitute the set $Y(c_i)$,; finally the weight $W(f)$ of each feature is updated. The update weight formula is:

$$W(f) = W(f) - \frac{\sum_{x \in X} diff(f, m_i, x_i)}{rd} + \frac{\sum_{c \neq \text{class}(m_i)} \left[\frac{P(C)}{1 - P(class(m_i))} \sum_{x \in Y(c_i)} diff(f, m_i, y_i)\right]}{rd}$$

where f denotes a certain feature, r denotes the number of samples, $diff(f, m_i, x)$ denotes the distance between sample X and sample Y with respect to a certain feature

f , P(C) denotes the proportion of C class samples to the total number of samples, and $class(m_i)$ denotes the category to which m_i belongs.

Dimensionality Reduction

SeqFea-Learn contains 15 dimensionality reduction methods.

Dimensionality Reduction Methods	Method Number
Principal component analysis (PCA)	1
Kernel PCA (KPCA)	2
Locally linear embedding (LLE)	3
Multi-dimensional scaling (MDS)	4
t-distributed stochastic neighbor embedding (T-SNE)	5
Truncated singular value decomposition (SVD)	6
Non-negative matrix factorization (NMF)	7
Gaussian random projection (GRP)	8
Sparse random projection (SRP)	9
Independent component analysis (ICA)	10
Factor analysis (FA)	11
Agglomerate feature (AF)	12
Autoencoder	13
Gaussian noise autoencoder	14
Variational autoencoder	15

1. Principal component analysis (PCA)

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. PCA aims to find the directions of

maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one. If we use PCA for dimensionality reduction, we construct a $d * k$ dimensional transformation matrix W that allows us to map a sample vector x onto a new k dimensional feature subspace that has fewer dimensions than the original d-dimensional feature space:

$$\begin{aligned} \boldsymbol{x} &= [x_1, x_2, \ldots, x_d], \quad \boldsymbol{x} \in \mathbb{R}^d \\ &\downarrow \boldsymbol{xW}, \quad \boldsymbol{W} \in \mathbb{R}^{d \times k} \\ \boldsymbol{z} &= [z_1, z_2, \ldots, z_k], \quad \boldsymbol{z} \in \mathbb{R}^k \end{aligned}$$

2. Kernel PCA (KPCA)

PCA is a linear method. That is it can only be applied to datasets which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.

3. Locally Liner Embedding (LLE)

The LLE algorithm, is based on simple geometric intuitions. Suppose the data consist of N real-valued vectors, each of dimensionality$D$, sampled from some smooth underlying manifold. We can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. In the simplest formulation of LLE, one identifies K nearest neighbors per data point, as measured by Euclidean distance. Reconstruction errors are then measured by the cost function:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

which adds up the squared distances between all the data points and their reconstructions.

4. Multi-dimensional Scaling (MDS)

Multidimensional scaling is a means of visualizing the level of similarity of individual cases of a dataset. MDS is used to translate “information about the pairwise distance among a set of n objects or individuals” into a configuration of n points mapped into an abstract Cartesian space.

It takes an input matrix dissimilarities between pairs of items and outputs a coordinate matrix whose configuration minimizes a loss function called strain. General forms of loss functions called stress in distance MDS and Strain in classical MDS. The strain is given by:

$$Strain_D(x_1, x_2, \ldots, x_N) = \left(\frac{\sum_{i,j} \left(b_{ij} - \langle x_i, x_j \rangle \right)^2}{\sum_{i,j} b_{ij}^2} \right)^{1/2}$$

where b_{ij} are terms of the matrix B defined on step 2 of the following algorithm.

Clustering method

SeqFea-Learn contains 7 clustering methods.

Clustering methods	Method Number
K-means	1
Spectral Clustering	2
Gaussian Mixture Clustering	3
Affinity Propagation Clustering	4
Mean Shift	5
DBSCAN	6
OPTICS	7

1. K-means clustering

K-means clustering is a method of vector quantization, originally from signal processing. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a porotype of the cluster. Given a set of observations (x_1,x_2,…x_n), where each observation is a d-dimensional real vector, k-means is to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i-1}^{k} |S_i| \operatorname{Var} S_i$$

where u_i is the mean of points in S_j. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster.

2. Spectral Clustering

   Spectral clustering techniques make use of the spectrum of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset.

Sampling method

SeqFea-Learn contains 5 sampling methods.

Sampleing methods	Method Number
Random over sampling (ROS)	1
Synthetic minority oversampling technique (SMOTE)	2
Adaptive synthetic (ADASYN)	3
Random under sampling (RUS)	4
Neighbourhood cleaning rule (NCR)	5

Classification method

SeqFea-Learn integrated 13 classification methods to make predictions.

Classifier	Method Number
Support vector machine (SVM)	1
K-nearest neighbor (KNN)	2
Random forest (RF)	3
Extremely randomized trees (Extra-Trees)	4
Gradient boosting decision tree (GBDT)	5
XGBoost	6
LightGBM	7
Bagging classifier (Bagging)	8
AdaBoost	9
Gaussian Naïve Bayes (GNB)	10
Deep neural network (DNN)	11
Convolutional neural network (CNN)	12
Recurrent neural network (RNN)	13

1. Support vector machine (SVM)

Support vector machine is a machine learning method based on statistical learning theory. It has better generalization ability and higher classification in solving nonlinear and high dimensional data problems. Its basic principle is to map the input sample set to the high-dimensional space, construct the optimal hyperplane, and automatically find out the support vectors that have better classification ability for the classification through the learning algorithm.

For a given sample dataset $(x_i, y_i), i = 1, 2, \cdots n, x \in R^d,\ y \in \{+1, -1\}$

$$y_i\left[\left(\omega^t x_i\right)+b\right]-1\geq 0, i=1,2,\cdots n$$

Then we can find the optimal classification plane. For the classification problem of high-dimensional space, the SVM model should be able to correctly classify the two types of samples while ensuring the maximum classification interval. Then using $k(x_i, x_j)$ kernel function to obtain the optimal classification plane:

$$f\left(x\right)=\operatorname{sgn}\left\{\left(\omega, x\right)+b\right\}=\operatorname{sgn}\left[\sum_{x_i\in sv}\alpha_i y_i k\left(x_i, x_j\right)+b\right]$$

The kernel functions in SVM include linear kernel functions, polynomial kernel functions, radial basis kernel functions, and sigmoid kernel functions.

2. K-nearest neighbor (KNN)

The idea of the KNN algorithm is to calculate the similarity of each sample between the test set and the training set. Then count the most similar samples to classify the prediction via the majority voting method. The specific steps of the KNN algorithm can be describe as:

Step 1: Input the sample dataset and calculate neighbor samples according to the feature information.

Step 2: calculate the sample weight in order, using the equation:

$$p(\bar{x}, C_j)=\sum_{\bar{d}_i\in KNN} Sim(\bar{x}, \bar{d}_i)y(\bar{d}_i, C_j)$$

where, $\bar{x}$ is the variable of the new sample, $Sim(\bar{x}, \bar{d}_i)$ is the similarity score and $y(\bar{d}_i, C_j)$ is the category attribute function.

Step 3 Compare the weights of the two categories and classify the samples the category with the largest weight.

3. Random forest (RF)

RF is a classifier based on decision tree and Bagging algorithm. First, the training sets are generated using the bootstrap method. Secondly, for each training set, the decision tree model is constructed. At each step of split selection, RF determines the optimal split node from a subset of features according to Gini coefficient. Then the ensemble learning method can be built up based on the majority voting principle

$$H(x)=\arg\max_{Y}\sum_{i=1}^{k} I(h_i(x)=Y)$$

where, h_i represents single decision tree, and Y is the output label.

4. Extremely randomized trees (ET)

ET is similar to RF, but RF applies the Bagging algorithm, and ET uses all training samples to construct each decision tree. Random forest obtains the optimal split attribute from a random subset, and the ET is completely random to obtain the split node.

5. XGBoost

XGBoost is an ensemble learning algorithm based on gradient boosting, which is an optimization model that combines a linear model with a boosting tree model. It uses not only the first derivative but also the second derivative of the loss function for second-order derivation.

For a given n sample and m feature datasets $D = \{(x_i, y_i)(|D| = n, x_i \in R^m, y_i \in R)\}$, the XGBoost algorithm objective function is defined as

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{t=1}^T \Omega(f_t)$$

where L is the loss function, the smaller L is, the better the performance of the algorithm. f_t is the t-th tree, $\Omega(f_t)$ is the regular term, used to control the complexity of the model, $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$, ω is the vector of the score in the leaf, λ is the regularization parameter, γ is the minimum loss required to further segment the leaf nodes. Then, according to Taylor expansion optimization of the objective function, the second-order Taylor of the loss function after the t iteration is obtained.

$$L^{(t)} \approx \sum_{i=1}^k \left[l\left(y_i, y_i^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x) \right) \right] + \Omega(f_t)$$

where $g_i = \partial_{y^{(t-1)}} l(y_i, y^{(t-1)})$ represents the first derivative of each sample and $h_i = \partial^2_{y^{(t-1)}} l(y_i, y^{(t-1)})$ represents the second derivative of each sample, and the loss function depends only on the first and second derivatives of each data point.

6. LightGBM

LightGBM is a variant of gradient tree boosting. As we can know, when the number of samples is large or the feature dimension is high, the efficiency and accuracy of GTB still cannot obtain satisfactory results. Ke et al proposed a highly efficient gradient boosting decision tree using gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) called LightGBM. This algorithm uses GOSS to determine the split point via calculating variance gain. First, sorting the absolute values of the gradients of the

training examples in descending order and the top $a \times 100\%$ data samples of gradient values are selected called A. Then the subset B whose size is $b \times |A^c|$ is randomly selected from the retained samples A^c. Finally, the instances are split through the estimated variance $V_j(d)$ on $A \cup B$.

$$V_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right)$$

where $A_l = \{x_i \in A: \ x_{ij} \le d\}$, $A_r = \{x_i \in A: \ x_{ij} > d\}$, $B_l = \{x_i \in B: \ x_{ij} \le d\}$, $B_r = \{x_i \in B: \ x_{ij} > d\}$, g_i represents the negative gradient of the loss function, $\frac{1-a}{b}$ is employed to normalize the sum of gradients.