Large-scale comparative review and assessment of computational methods for anti-cancer peptide identification.

In this work, we comprehensively investigate 16 state-of-the-art predictors for ACPs in terms of their core algorithms, feature encoding schemes, performance evaluation metrics and webserver/software usability. An ensemble learning framework, termed ACPredStackL, is employed to evaluate the performance of existing predictors for ACPs. Comprehensive performance assessment is conducted to evaluate the robustness and scalability of the existing predictors using a well-prepared benchmark dataset.

Requirements

The following python packages are required when building the stacking model.

==========

Mlxtend>=0.16
Sklearn>=0.0
Lightgbm>=2.2.3
Xgboost>=1.2.1

==========

*i*Learn [1] source package is required to be added to your project. The package is used to handle several pre-processing operations for feature engineering including feature extraction, feature standardization, feature selection, etc.

ACPredStackL is a stacking ensemble learning model to predict ACPs. The core coding is to build the ACPredStackL model using 'build_model' in the 'model.py'. Similar to other classification models, it can be used to build a classifier to predict ACPs from protein or peptide sequences. Once an ensemble stacking classifier is built, the method named with 'fit ()' is called to train the model on the training dataset, and the method named with 'predict ()' to identify ACPs from the test dataset. The main steps are as follows.

Step 1. Feature extraction

You use *i*Learn package to extract five sequence encoding schemes including AAC、PAAC、CTD、CKSAAP and QSOrder to represent input peptide sequences. The command lines are as followings.

```
cmd1 = "python iLearn-master/descproteins/PAAC.py --file " + file + " --lamada 4 --format csv
--out " + dir + "/PAAC.csv"

cmd2 = "python iLearn-master/iLearn-protein-basic.py --file " + file + " --method CTDD --format
csv --out " + dir + "/CTDD.csv"
```

```
cmd3 = "python iLearn-master/descproteins/QSOrder.py --file " + file + "    --lag 5 --format csv
--out " + dir + "/QSOrder.csv"

cmd4 = "python iLearn-master/descproteins/CKSAAP.py --file " + file + "    --gap 4 --format csv
--out " + dir + "/CKSAAP.csv"

cmd5 = "python iLearn-master/iLearn-protein-basic.py --file " + file + " --method AAC    --format
csv --out " + dir + "/AAC.csv"
```

Step 2. Feature standardization

You can use z-score to standardize features using the sklearn package. Alternately, ZScore implemented in *i*Learn package can be used to perform standardization of features.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(data)
scaler.transform(data)
```

Step3. Feature selection

You can first use Fscore algorithm implemented in the *i*Learn package to obtain feature rank list, and then build a stacking ensemble learning model by calling the method 'build_model' in the model.py. The sequential forward search procedure (FSP) [2] is employed to select the suboptimum subset of features with stronger identification ability of ACPs.

```
"python iLearn-master/iLearn-feature-selectior.py --file " + filename + " --method Fscore --format
csv --out "+ featureRank
```

Step 4. *k*-fold cross-validation test and independent test

Similar to other classification model, once a stacking model is built, you can perform the *k*-fold cross validation test and independent test. The method 'fit()' is called to perform *k*-fold cross validation test on the training dataset, and 'predict ()' is called to perform the independent test.

```
from model import build_model

acp_model = build_model()
acp_model.fit()
acp_model. Predict()
```

References:

1. Chen Z, Zhao P, Li F et al. *i*Learn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data, Briefings in Bioinformatics 2020;21:1047-1057.
2. Whitney AW. A Direct Method of Nonparametric Measurement Selection, IEEE Transactions on Computers 1971;20:1100-3.