Supplementary Material

LipoSVM: Prediction of Lysine Lipoylation in Proteins based on the Support Vector Machine

Meiqi Wu[1], Pengchao Lu[2], Yingxi Yang[3], Liwen Liu[1], Hui Wang[4], Yan Xu[1] and Jixun Chu[1,*]

[1]Department of Applied Mathematics, University of Science and Technology Beijing, Beijing 100083, China; [2]Equipment Leasing Company of China Petroleum Pipeline Engineering Co., Ltd. 065000 Langfang City, Hebei Province, China; [3]Department of Chemical and Biological Engineering, Hong Kong University of Science and Technology, Hong Kong, China; [4]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China

Supplementary S3. Detailed process of constructing PSSM.

To get information of sequential evolution, the position-specific scoring matrix [1] can be utilized. Let *P* and *N* represent the flaking regions of positive and negative samples, N^+ and N^-represent the number of fragments in the positive and negative dataset, respectively. P_i is the i–th fragment in the positive dataset, P_{ij} is the j–th position in the i–th fragment. Then $V_P^{j,a}$ which is a binary vector represents each symbol a from the 21 amino acids and each position in *P*.

$$V_P^{j,a} = (A_1, A_2, \cdots, A_{N^+}) \quad (1)$$

where A_i can be calculated as following:

$$A_i = \begin{cases} 1 & P_{ij} = a \\ 0 & P_{ij} = a \end{cases}. \quad (2)$$

For negative samples the vector $V_N^{j,a}$ can be formed in the same way. A *p*-value for each set of $V_P^{j,a}$ and $V_N^{j,a}$ obtained via two-sample *t*-test [2]. Then, the following matrix V_{PSSM} constructed.

$$V_{PSSM} = \begin{bmatrix} V_{1,1} & V_{1,2} & \cdots & V_{1,L} \\ V_{2,1} & V_{2,2} & \cdots & V_{2,L} \\ \vdots & \vdots & \vdots & \vdots \\ V_{21,1} & V_{21,2} & \cdots & V_{21,L} \end{bmatrix} \quad (3)$$

In this matrix, L is the length of the fragments. $V_{i,j}$ denotes the *p*-value of the i–th amino acid in the j–th position for a given positive and negative dataset. By calculating the frequency of each amino acid in each position of the positive dataset, the following matrix F^Pobtained.

$$F^P = \begin{bmatrix} F_{1,1}^P & F_{1,2}^P & \cdots & F_{1,L}^P \\ F_{2,1}^P & F_{2,2}^P & \cdots & F_{2,L}^P \\ \vdots & \vdots & \vdots & \vdots \\ F_{21,1}^P & F_{21,2}^P & \cdots & F_{21,L}^P \end{bmatrix} \quad (4)$$

where $F_{i,j}^P$ represents the frequency of the i–th amino acid in the j–th position and as is F^N. Finally, the following PSSM matrix is used for encoding.

$$M_{PSSM} = \begin{bmatrix} E_{1,1} & E_{1,2} & \cdots & E_{1,L} \\ E_{2,1} & E_{2,2} & \cdots & E_{2,L} \\ \vdots & \vdots & \vdots & \vdots \\ E_{21,1} & E_{21,2} & \cdots & E_{21,L} \end{bmatrix} \quad (5)$$

where $M_{i,j}$ can be calculated as following:

$$\delta_{i,j} = \frac{F_{i,j}^P - F_{i,j}^N}{V_{i,j}} \quad (6)$$

$$M_{i,j} = \begin{cases} \ln(|\delta_{i,j}| + 1) & \delta_{i,j} \geq 0 \\ -\ln(|\delta_{i,j}| + 1) & \delta_{i,j} < 0 \end{cases}. \quad (7)$$

If $M_{i,j}$>0, the probability that the i–th amino acid in the j–th position appears in the positive fragments is greater. Otherwise, it is more likely to be in the negative fragments.

REFERENCES

1. Hasan, M. A. M.; Ahmad, S.; Molla, M. K. I., iMulti-HumPhos: a multi-label classifier for identifying human phosphorylated proteins using multiple kernel learning based support vector machines. *Mol Biosyst* **2017,** *13* (8), 1608-1618.
2. Vacic, V.; Iakoucheva, L. M.; Radivojac, P., Two Sample Logo: a graphical representation of the differences between two sets of sequence alignments. *Bioinformatics* **2006,** *22* (12), 1536-7.

Supplementary S4. Pseudo codes of SMOTE algorithm.

Pseudo code of SMOTE algorithm

Algorithm SMOTE (*T, N, k*)
Input: Number of minority class examples *T*; Amount of SMOTE *N*%; Number of nearest neighbors *k.*
1. (If *N* is less than 100%, randomize the minority class sample as only a random percent of them will be SMOTEd.)
2. if $N < 100$
3. then Randomize the *T* minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. end if
7. $N = (int)N/100$
8. *k* = Number of nearest neighbors
9. *numattrs* = Number of attributes
10. *Sample* [][]: array for original minority class samples
11. *newindex*: keeps a count of number of synthetic samples generated, initialized to 0.
12. *Synthetic* [][]: array for synthetic samples
(Compute *k* nearest neighbors for each minority class sample only.)
13. for $i = 1 : T$
14. Compute *k* nearest neighbors for *i,* and save the indices in the nnarray
15. Populate (*N, i, nnarray*)
16. end for

Populate (*N, i, nnarray*)
17. while $N \neq 0$
18. Choose a random number between 1 and *k*, call it *nn*. This step chooses one of the *k* nearest neighbors of *i*.
19. for *attr*=1: *numattrs*
20. *dif* = *Sample*[*nnarray*[*nn*]][*attr*]-*Sample*[*i*][*attr*]
21. *gap* = random number between 0 and 1
22. *Synthetic*[*newindex*][*attr*] = *Sample*[*i*][*attr*]+*gap***dif*
23. endfor
24. *newindex++*
25. *N=N-1*
26. endwhile

Output: (*N/100*) * *T* synthetic minority class samples