De novo profile generation based on sequence context specificity with the long short-term memory network

Kazunori D Yamada[1,2] and Kengo Kinoshita[1,3,4*]

[1]Graduate School of Information Sciences, Tohoku University, Sendai, Japan, [2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan, [3]Tohoku Medical Megabank Organization, Tohoku University, Sendai, Japan, [4]Institute of Development, Aging, and Cancer, Tohoku University, Sendai, Japan

ABSTRACT

Amino acid sequence profiles are widely used for bioinformatics studies, such as sequence similarity searches, multiple alignments, and evolutionary analyses. Currently, many biological sequences are becoming available, and the rapidly increasing amount of sequence data emphasizes the importance of scalable generators of amino acid sequence profiles. We employed a long short-term memory (LSTM) network and developed a novel profile generator to construct profiles without any assumptions, except for input sequence context. Our method could generate better profiles than existing de novo profile generators, including CSBuild and RPS-BLAST on the basis of profile-sequence similarity search performance with linear calculation costs against input sequence size. In addition, we analyzed the effects of the memory power of LSTM and found that LSTM had high potential power to detect long-range interactions between amino acids, as in the case of beta strand formation, which has been a difficult problem in protein bioinformatics using sequence information.

INTRODUCTION

Amino acid sequence profiles or position specific scoring matrices (PSSMs) are matrices in which each row contains evolutionary information regarding each site of a sequence. PSSMs have been widely used for bioinformatics studies, including sequence similarity searches, multiple sequence alignments, and evolutionary analyses. Additionally, modern sequence-based prediction methods of protein properties by machine learning algorithms often use PSSMs derived from input sequences as input vectors of the prediction. A PSSM is typically constructed by iterative searches of a query sequence against a huge sequence database, such as nr or UniProt [1]. The iteration is a type of machine learning process that improves the quality of profiles gradually. In recent years, the most successful profile generation method was HHBlits [2]. HHBlits generates profiles by iterative searches of huge sequence databases, as in the case of PSI-BLAST [3]; however, HHBlits uses the hidden Markov model (HMM) profile, whereas PSI-BLAST adopts PSSM. To the best of our knowledge, these methods can produce good profiles on the basis of the performance of similarity searches, but require an iterative search of a query sequence; therefore, the profile construction time depends on the size of the database. The recent increase in available biological sequences has made it more difficult to construct profiles.

In this context, de novo profile generators, such as CSBuild [4, 5] and RPS-BLSAT (DELTA-BLAST) [6], have been developed in order to reduce the cost of profile generation. CSBuild internally possesses a 13-mer amino acid profile library, which is a set of sequence profiles obtained by iterative searches of divergent 13-mer sequences. CSBuild searches short profiles against the short profile library for every part of a sequence and subsequently constructs a final profile for the sequence by merging the short profiles. CSBuild can reduce the profile construction time using precalculated short profiles; however, there is no theoretical evidence demonstrating that a PSSM can be constructed by integrating patchworks at the short (13-mer) sequence window. In other words, the previous study assumed that the protein sequences had a short context-specific tendency for the residues. This is also the case with RPS-BLAST, in which a batch of profiles obtained by searches of a query sequence against a precalculated profile library is assembled to construct a final profile.

Recently, neural networks have attracted increasing attention from various research areas, including bioinformatics. Neural networks are computing systems that mimic biological nervous systems of animal brains. Theoretically, if a proper activation function is set to each unit in middle layer(s) of a network, it can approximate any function [7]. In recent years, neural networks have been vigorously applied to bioinformatics studies. In particular, deep learning algorithms are typically applied to neural networks. For example, several studies have applied deep learning algorithms to predict protein-protein interactions [8, 9], protein structures [10, 11], residue contact maps [12], and backbone angles and solvent accessibilities [13]. The successes of deep learning algorithms have been realized by complex factors, such as recent increases in available data, performance improvements of semiconductors, development of optimal activation functions [14], and optimization of gradient descent methods [15]. These various factors have

*To whom correspondence should be addressed. Tel: +81 22 795 7179; Email: kengo@ecei.tohoku.ac.jp

© 2017 The Author(s)
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/2.0/uk/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

enabled calculations that were thought to be infeasible, and modern deep learning algorithms now not only stack the layers of multilayer perceptrons but also generate various types of inference methods, including stacked autoencoders, recurrent neural networks (RNNs), and convolutional neural networks [14].

The RNN is one of the most attractive deep learning methods. More specifically, long short-term memory (LSTM) [16], an RNN, can be a promising method for learning the time series or context of input vectors. Namely, with LSTM, it may be possible to learn an amino acid sequence context to predict the internal properties of amino acids sequences. The memory of LSTM is experimentally confirmed to be able to continue for more than 1,000 time steps, though theoretically it can continue forever [16]. This memory power may be sufficient to learn features from protein sequences, for which lengths are generally less than 500. In addition, compared with window-based prediction methods, we do not need to assume that some protein internal properties, such as secondary structure, steric structure, or evolutionary information, are formed in some lengths of amino acid sequences, as in the case of CSBuild, which assumes 13-mers. LSTM can even learn such optimal lengths of context automatically throughout learning. This characteristic of LSTM is thought to be more suitable for protein internal property predictions. Indeed, several machine learning based prediction methods utilizing LSTM network for protein property prediction have been successful applied [13, 17, 18].

In this study, we attempted to develop a de novo profile generator that mimicked the ability of the existing highest performance profile generation method, HHBlits, using an LSTM network, expecting our generator to be able to include the ability to input whole protein sequences. In addition, we analyzed the importance of sequence context in the prediction and performance of LSTM to solve specific biological problems through our computational experiments.

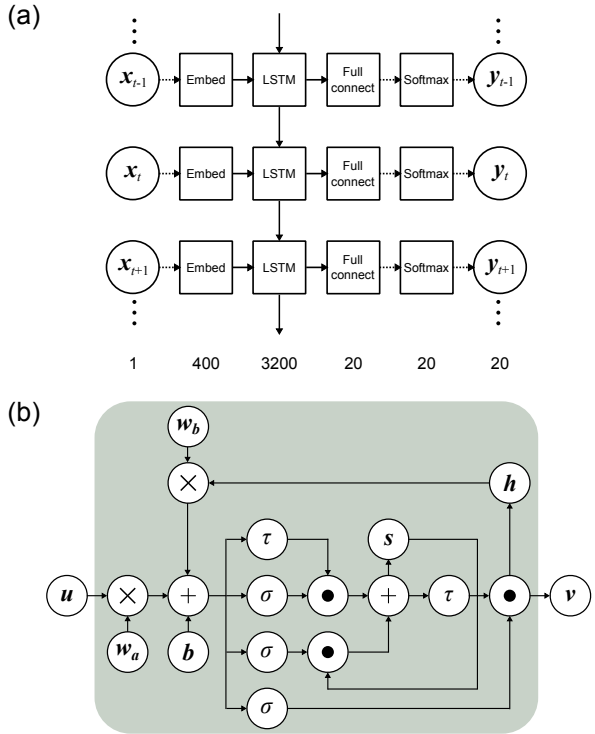

Figure 1. Network of learning. (a) Overview of the designed network in this study. Here, $\mathbf{x}$, $\mathbf{y}$ and t represent an input vector, an output vector and a position of an amino acid sequence. In the squares, "Embed", "Full connect", "Softmax" stand for a word embedding operation, a fully connected network, and a softmax function layer, respectively. The solid and broken arrows represent a matrix operation and an array operation, respectively. The numbers at the bottom of panel (a) stand for a dimension of vectors of each layer. (b) Description of LSTM layer. Here, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{h}$, $\mathbf{s}$, $\times$, $+$, dot, τ, σ, $\mathbf{w_a}$, $\mathbf{w_b}$ and $\mathbf{b}$ stands for an input vector to LSTM unit, an output vector from LSTM unit, a previous input vector, an unit for constant error, a multiplication of matrices, a summation of matrices, a Hadamard product calculation, a hyperbolic tangent, a sigmoid function, a weight matrix to be learned, another weight matrix and a bias vector.

METHODS

Learning dataset

We conducted iterative searches using HHBlits version 2.0.15 with the default iteration library provided by the HHBlits developer and generated profiles of the sequences in Pfam40 [19], where the maximum percent identity for all pairs of sequences was less than 40%. Because we used the SCOP20 test dataset as a benchmark dataset for the performance of profile generators (see below), we excluded all sequences sharing homology with any sequence in the SCOP20 test dataset from the Pfam40 dataset using gapped blast (blastpgp) searches prior to the iterative search, where we considered an e-value of less than 10^{-10} as a homologous hit. The number of HHBlits iterations was set to three. Although HHBlits produces HMM profiles, we converted these profiles to PSSMs by extracting amino acid emission frequencies of match states. Finally, we set the generated profiles as target vectors and its corresponding sequences as input vectors in learning steps. Namely, in our learning scheme, each instance included an N dimension vector (sequence) as an input vector and a $20 \times N$ dimension vector (profile) as a target vector, where N represents sequence length, and 20 is the number of types of amino acid residues.

Learning network

We designed a network with an LSTM layer, as shown in Figure 1a. In the learning steps, each amino residue in the input sequence was converted to a 400-dimension float vector by a word embedding method [20]. After the word embedding process, the input vectors were processed by an LSTM layer followed by a fully connected layer. The output of the network was set to a solution of the softmax function of the immediately anterior layer. We set the unit size of each gate of the LSTM unit to 3,200. As a cost function, we used the root mean square error between an output of the network and a target vector. As an optimizer of the gradient descent method, Adam was used [15]. As an LSTM unit, we utilized an extended LSTM with a forget gate [21], as shown in Figure 1b. In Figure 1b, the top, middle, and bottom sigmoid gates represented the input, forget, and output gates, respectively. For regularization, we used a dropout method

against weights between an input layer and an LSTM layer with a drop ratio of 0.5. We observed learning and validation curves to avoid overfitting and stopped learning steps at 5,000 epochs. Because we could not deploy whole sequence data into the memory space in our computational environment, we randomly selected 40,000 sequences (about 1/40th of all sequences) and learned them as a one epoch. Therefore, an epoch in this study was about 40 times the typical epoch.

As a framework to implement the learning network, we used Chainer version 1.15.0.1 (Preferred Networks) with CUDA and cuDNN version 6.5 (NVIDIA), and the calculations were performed by a server with Tesla K20m (NVIDIA) at the NIG supercomputer at ROIS National Institute of Genetics in Japan.

Benchmark of the performance of similarity searches

Performances of profile generators were evaluated based on the results of similarity searches with their generated profiles. As representatives of existing methods of rapid profile generators, we compared our method with CSBuild version 2.2.3 and RPS-BLAST version 2.2.30+. As a test dataset, the SCOP20 test dataset was used, as in the original paper for CSBuild [4], which consists of 5,819 sequences with protein structural information; the maximum percent identity of the sequences in the dataset was less than 20%. As a profile library for CSBuild, the data from the discriminative model of CSBuild (K4000.crf) were used. For RPS-BLAST, we excluded all entries sharing homologous relationships with the SCOP20 test dataset from the conserved domain database for DELTA-BLAST version 3.12 using the same method as that used to make the learning dataset.

To eliminate any biases of alignment algorithms, all profiles in this study were converted to the PSI-BLAST readable format and used as input files in a PSI-BLAST search. As an application of PSI-BLAST, we used blastpgp version 2.2.26 for CSBuild, since CSBuild outputs blastpgp-readable profile files. For the other methods, psiblast version 2.2.30+ was used. There were no significant differences in sensitivity or similarity searchers between these two versions of PSI-BLAST (data not shown). The results of the similarity searches were sorted according to their statistical significance in descending order. Each hit was labeled as a true positive, false positive, or unknown based on the evaluation ruleset for SCOP 1.75 benchmarks [22]. With this information, we described the receiver operating characteristic (ROC) curves and evaluated the performance [23]. As an evaluation criterion, we used the AUC1000 value, which corresponds to the area under the ROC curve until 1,000 false positives were detected.

The profile generation time was benchmarked on an Intel(R) Xeon(R) CPU E5-2680 v2 @2.80 GHz with 64GB RAM using a single thread.

RESULTS AND DISCUSSION

Training a predictor with LSTM

In this study, we assumed profiles generated by HHBlits as ideal profiles and used these as target profiles in training steps. We then attempted to generate profiles as similar to the HHBlits profiles as possible with a predictor using LSTM.

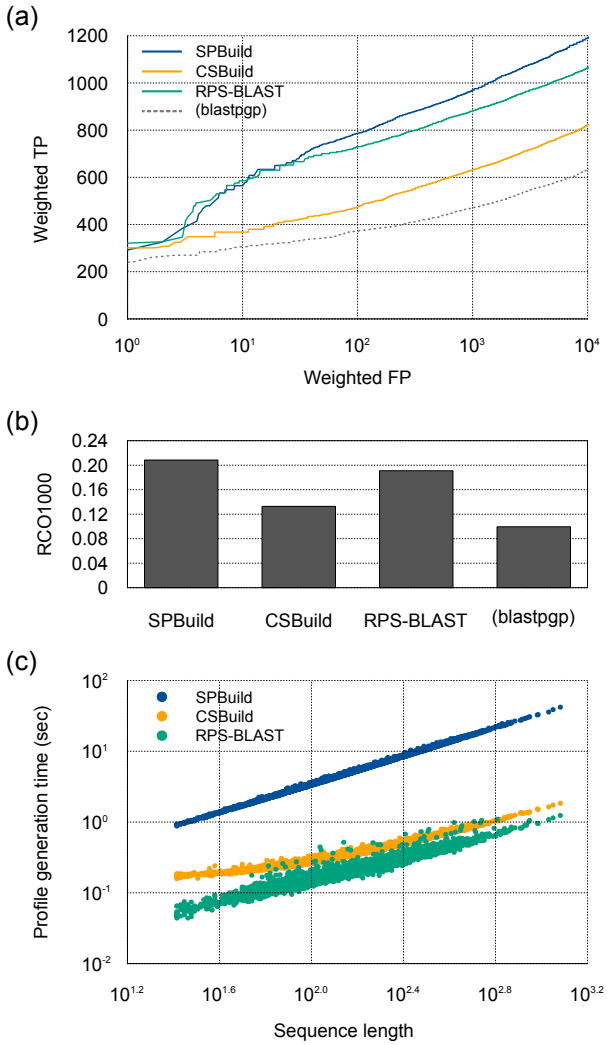

Figure 2. Performance comparisons of (a, b) similarity searches and (c) calculation time. (a) ROC curves of SPBuild and other methods. Here, the performance of blastpgp was added for a reference. (b) AUC 1000 values of SPBuild, CSBuild, RPS-BLAST, and blastpgp. (c) The scatterplot of the profile generation time for each method on the SCOP20 test dataset.

The performances of similarity searches with the profiles generated by HHBlits were better than those of the other methods [2].

Initially, we selected amino acid sequences with lengths of 50–1,000 in Pfam40, sequences without irregular amino acid characteristics, and those sharing homology with the SCOP20 test dataset. We also included 1,329 sequences derived from the SCOP20 learning dataset [4] to the final learning dataset for our reference. As a result, we obtained 1,602,338 sequences and calculated a profile using HHBlits for each sequence. With this training dataset, we trained the predictor shown in Figure 1a. For learning, we used 20,000 randomly extracted instances as a validation dataset and checked whether the predictor overfit the training dataset. The number of mini-batches was set to 200, and each amino

acid was converted to a 400-dimension float vector by the word embedding method, as described in section 2.2. For each sequence, the starting site of learning was not confined to the N-terminal but was selected at random in order to avoid overfitting of the predictor to the specific site. We observed learning and validation curves to confirm the lack of overfitting and stopped learning at 5,000 epochs (Figure S1). Even using GPU computing, completion of our calculations required almost two months.

Using the obtained parameters (weight matrices and bias vectors through the learning), we constructed a novel de novo profile predictor, which we called Synthetic Profile Builder (SPBuild). Our profile generator can be downloaded from http://yamada-kd.com/product/spbuild.html.

Performance comparisons

First, we compared the performance of similarity searches of profile generators. The profiles for all sequences in the SCOP20 test dataset were generated by each method, and all-against-all comparisons of the test dataset by PSI-BLAST with the obtained profiles were conducted. As profile generators, we evaluated the de novo profile generators CSBuild and RPS-BLAST, in addition to SPBuild. We also added the performance of PSI-BLAST without iterations (blastpgp) as a representative sequence-sequence based alignment methods for reference. In addition, HHBlits was further compared as another reference, and the results are shown in Figure S2.

As shown in Figure 2a, CSBuild was clearly superior to the sequence-sequence based alignment method, blastpgp, and RPS-BLAST, and SPBuild showed slightly better performance. When performance was evaluated by AUC1000 values (Figures 2b and S2a), the values of our method, CSBuild, RPS-BLAST, and HHBlits were 0.208, 0.133, 0.191, and 0.439, respectively. Notably, the performance of our method (0.208) did not reach that of HHBlits (Figure S2a, AUC1000 = 0.439), even though we trained our predictor with outputs of HHBlits, indicating that SPBuild was not completely able to mimic the ability of HHBlits.

This tendency was also true for another benchmark result, where we evaluated the performance of SPBuild and HHBlits on the SCOP20 learning dataset instead of the test dataset (Figure S2b). Our findings were surprising because the SCOP20 learning dataset was a part of the learning dataset for construction of the predictor with LSTM, and the performance of our predictor should reach that of HHBlits. One possible reason for the observation is that LSTM may not have worked properly on our learning scheme. To examine this possibility, we performed another learning method to examine the performance of LSTM itself with our learning scheme, where we trained a predictor with only the SCOP20 learning dataset and let the predictor overfit the dataset. As a result, the performance of the predictor was almost the same as that of HHBlits, as expected (Figure S2c). This result indicated that LSTM could precisely learn input sequence properties and output correct PSSMs, but that the performance of the predictor was worse than that of SPBuild with proper learning due to the overfitting of the predictor to the learning dataset (Figure S2d). In short, these results suggested that LSTM correctly worked and that relationship between performance and overfitting was a simple trade-off. Therefore, we concluded that SPBuild could be trained moderately without conflict under our learning dataset and hyperparameters.

Next, we evaluated the profile generation time of each method. Table 1 shows the mean computation time of profile generation using the SCOP20 test dataset. As a result, SPBuild was almost 20 times faster than HHBlits, although CSBuild and RPS-BLAST were still faster than SPBuild. However, we think the most important property of a sequence handling method in the big data era is scalability to the data, namely time complexity of the method against the input sequence length. Theoretically, the time complexity of our method would be linear compared with the input sequence length, similar to CSBuild and RPS-BLAST. To clarify this point, we plotted profile generation times (seconds) versus input sequence lengths (N), as shown in Figure 2c. When the instances were fitted to a line, the determination coefficient was 0.998, and the slope of the line was 1.00. This result indicated that the time complexity of our method was $O(N)$. Notably, the slopes of CSBuild and RPS-BLAST appeared to be less than 1.0 in the figure; however, errors in the experiments or other factors in the implementation of these programs may have caused this because the costs of these calculations must be higher than that of $O(N)$. Although our method required much time to compute large matrix calculations in the neural network layers and was therefore slower than CSBuild and RPS-BLAST with the currently used sequence database, but our method had linear scalability against the number of input sites or sequence length and the number of input sequences.

Memory power of LSTM in our problem

We also examined the memory power of LSTM in our problem in order to determine the feasibility of the LSTM approach for sequence-based predictions. For this purpose, we considered the reset time lengths of memory cells ($\mathbf{h}$ in Figure 1b) at sequence lengths of 5, 10, 20, 30, 50, 100, 200, and 300 and for full-length sequences. We then benchmarked the performances of similarity searches with the SCOP20 test dataset. The memory reset time length was directly linked to the memory power of the predictors, and a predictor with a memory reset time length of 5, for example, generated profiles based on information from the previous five sites, including the current site. As a result, the performance of similarity searches clearly changed as the memory power decreased (Figure 3a). We also checked the performance of CSBuild with the same plot (Figure 3a). As described above, CSBuild constructs profiles by merging 13-mer short profiles; thus, we

Table 1. Comparison of profile generation times.

	Mean	SD
SPBuild	5.99	3.83
CSBuild	0.390	0.161
RPS-BLAST	0.233	0.120
HHBlits	120	105

Means and standard deviations (SDs) of profile generation times (s) against 5,819 sequences in the SCOP20 test dataset.

imagined that its performance would be similar to that of the LSTM profile predictors with low memory power. However, we found that the performance of CSBuild was located in the middle between memory powers of 30 and 50 for the LSTM predictors. We are not sure why this happened, but it may be because the sensitivities (corresponding to vertical axis of Figure 3a) of LSTM predictors were worse than expected or because of the excellence of CSBuild implementations.

To improve our understanding of the generated profiles by SPBuild, we evaluated the mean prediction accuracy (cosine similarity between output vector, y, and target vector) of SPBuild for each position of a residue on whole input sequences and observed that there was a clear transition in the plot (Figure 3b). The prediction accuracy of the initial portion (∼50) was worse than those of the other parts. This lower performance could be caused by the nature of LSTM. LSTM initializes the internal state of memory (**h**) by a null vector, which does not reflect any features of the learning dataset; thus, the prediction would be not stable until LSTM memorizes and stores a certain level of context information into memory. In our case, the level of context information was 50–60 residues. Additionally, the decrease in accuracy in the last part (∼200) was derived from the nature of our learning dataset; the mean length of SCOP20 was about 154, and SPBuild may be able to be optimized for the average length. This consideration was consistent with the observation that improvement of the performance with memory power of 200 and 300 decreased compared with smaller memory power lengths (Figure 3a). In conclusion, these results suggested that substantially long length context, ideally speaking, the context of the sequence length or at least more than 50, would be required to predict precise profiles. Protein primary and secondary structures, including solvent accessibility and contact number, must be restricted by protein steric structures, which are formed by complex remote interactions of amino acid residues. Our findings reflect the influence of remote relationships stemmed from the steric structure on sequence context. In other words, LSTM will be a powerful predictor for divergent features of proteins, if appropriate memory power length is used. Indeed, other sequence-based predictors using LSTM have achieved successful outcomes and shown the high feasibility of LSTM [13, 17, 18].

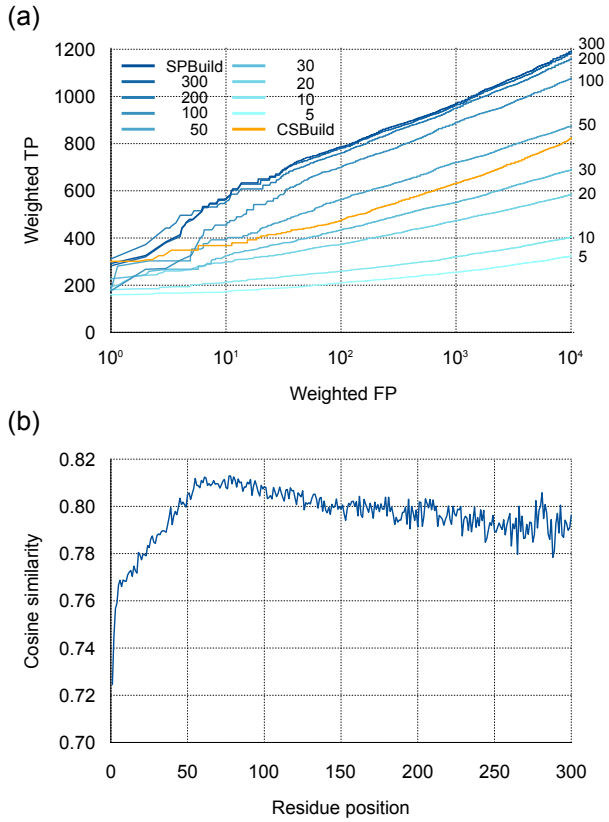

Figure 3. Effects of memory power of LSTM on predictors. (a) Comparison of profile generators with various reset lengths of memory on LSTM. The benchmark dataset was the SCOP20 test dataset. The reset time of SPBuild corresponded to the input sequence length. (b) Mean cosine similarity between output vectors of SPBuild and target vectors as a function of the position of residues in input sequences of the SCOP20 test dataset.

Long-range interactions and memory lengths

As shown in Figure 4a, we calculated the AUC1000 values of SPBuild relative to those of CSBuild and RPS-BLAST for each SCOP class. The values were calculated by dividing the AUC1000 value of SPBuild by that of each method, which indicated how the sensitivity of SPBuild was better than those of the existing methods for each SCOP class. Actual AUC1000 values are shown in Table S1. Notably, the performance of SPBuild was 2.03- and 1.29-fold higher than those of CSBuild and RPS-BLAST for SCOP class b, respectively. SCOP class b consists of β proteins. Generally, β-strands are constructed by remote interactions between residues when compared with α-helices. Secondary structure predictors with a window-based method developed by machine learning methods tend to show poorer performance in β regions than in α regions. The main reason for this weakness is related to the long-range interactions in β structures, which may not be properly handled by the limited lengths of sequence windows [24, 25]. This tendency may be also observed with the profile predictors. CSBuild constructs final profiles by assembling short window-based profiles, and RPS-BLAST also combines many subjected profiles obtained by local similarity searches against profile libraries. The actual mean length of the profiles evaluated by RPS-BLAST with three iterations (default) on the SCOP20 test dataset was 77, which was relatively longer than that of CSBuild but still shorter than the typical length of a protein. However, our method can theoretically memorize whole-length amino acid sequences and can take the remote relationship into consideration to generate profiles.

To confirm the relationship between memory power length and structural categories, we calculated relative sensitivities for different reset time lengths (Figure 4b and 4c). As a result, the performance improvements in the b category were much better than those other categories, indicating that memory power was the most important factor for encode long-range interactions, such as β structures.

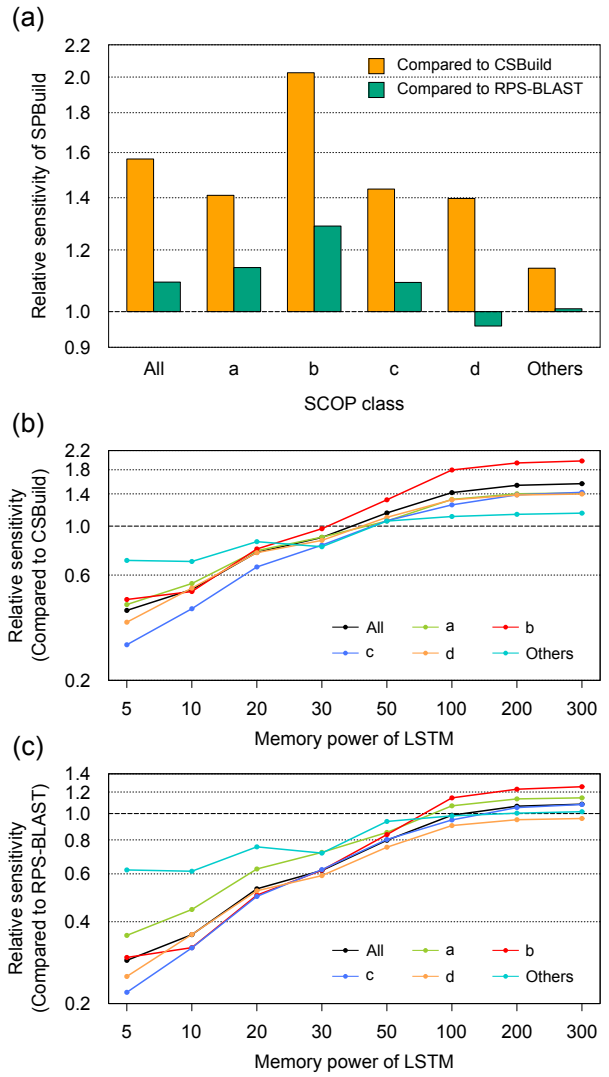

Figure 4. Relative sensitivity of SPBuild against existing methods on the test dataset. (a) The relative sensitivity of SPBuild against existing methods was calculated by dividing the AUC1000 of SPBuild by that of each method. Here, the label "others" includes SCOP classes e, f, and g. (b) The relative sensitivity of the profile generator with various memory powers of LSTM against CSBuild. (c) The relative sensitivity of the profile generator with various memory powers of LSTM against RPS-BLAST.

Limitation of SPBuild

As described, our method could generate profiles faster than HHBlits and showed higher performance than CSBuild and comparable to or slightly higher performance than RPS-BLAST, particularly for β region prediction, possibly due to the memory effects of LSTM. However, there are still some limitations to this method.

One of the limitations of SPBuild is the profile generation time, although the time complexity is linear against input sequence length. SPBuild used huge parameters, particularly for the LSTM layer, to calculate the final profile prediction. Although we set the size of the parameters to the current scale in order to maximize the final performance of SPBuild, we may be able to reduce the size and improve the calculation time if we are able to find more efficient network structures to learn amino acid context. In other words, to resolve the problem, exhaustive optimization of the hyperparameters of LSTM and/or development of novel network structures will be required.

The performance of iteration search with profiles made by de novo profile generators would be another interesting point for users. To check the performance of iteration searches, we calculated ROC curves for SPBuild, CSBuild, and RPS-BLAST and found that differences in performance became more unclear as the number of iterations increased (Figure S3). The result suggested that the performance of the initial search, or qualities of profiles, would be of small importance for the final results in iterative searches, if a sufficient number of iterations was used. The reason for this result is unclear; however, we believe that homologous sequences in the sequence space are limited and that almost all homologous sequences can be detected by using modestly good profiles, if a large number of iterations is used. Based on the sensitivity of profile-sequence-based similarity searches, our method may not be too attractive; however, there are many other uses for profiles. For example, profile-profile similarity searches, where profiles are generated by iterative searches of whole datasets, will be candidates for application of our approach. The bottleneck of profile-profile searches may be easily resolved with the rapid profile generator. In addition, profiles are often used to encode amino acids into input vectors in other machine learning methods. Machine learning methods generally require large learning data, and at this time, long-time iterative searches should be avoided because the calculation time increases depending on the learning data size. In such cases, higher speeds and accurate profile generators will be quite useful.

CONCLUSION

In this study, we developed a novel de novo generator of PSSMs using a deep learning algorithm, the LSTM network. Our method, SPBuild, improved the performance of homology detection with a rapid computation time compared with existing de novo generators. However, our goal was not to just provide an alternative method for profile generator but also to elucidate the importance of sequence context and the feasibility of LSTM for overcoming the sequence-specific problem. Our analyses demonstrated the effectiveness of memories in LSTM and showed that SPBuild achieved higher performance, particularly for β-region profile generation, which was difficult to predict by window-based prediction methods. This performance could be explained by the fact that our method utilized the LSTM network, which could capture remote relationships in sequences. Additionally, further analyses suggested that substantially long context was required for correct profile generation. This finding may be useful for the development of other prediction methods.

Profiles are the most fundamental data structures and are used for various sequence analyses in bioinformatics studies. Using SPBuild, the performance of sophisticated comparison algorithms, such as profile-profile comparison methods and multiple sequence alignment, can be further improved. In addition, profiles generated by SPBuild can be useful as input

vectors for other machine-based meta-predictors of protein properties.

ADDITIONAL INFORMATION

Acknowledgements

We are grateful to Kentaro Tomii and Toshiyuki Oda for constructive discussion. Computations were partially performed on the NIG supercomputer at ROIS National Institute of Genetics and the supercomputer system Shirokane at Human Genome Center, Institute of Medical Science, University of Tokyo.

Funding

This work was supported in part by the Top Global University Project from the Ministry of Education, Culture, Sports, Science, and Technology of Japan (MEXT) and the Platform Project for Supporting in Drug Discovery and Life Science Research (Platform for Drug Discovery, Informatics, and Structural Life Science) from the Japan Agency for Medical Research and Development (AMED).

Availability of data and material

The source code of SPBuild are available at http://yamada-kd.com/product/spbuild.html.

Abbreviations

LSTM: long short-term memory; PSSM: position-specific scoring matrix

Competing interests

The authors declare that they have no competing interests.

REFERENCES

1. Resource Coordinators NCBI. Database resources of the national center for biotechnology information. *Nucleic acids research*, 45(D1):D12, 2017.
2. Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature methods*, 9(2):173–175, 2012.
3. Stephen F Altschul, Thomas L Madden, Alejandro A Schffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25:3389–3402, September 1997.
4. Christof Angermüller, Andreas Biegert, and Johannes Söding. Discriminative modelling of context-specific amino acid substitution probabilities. *Bioinformatics*, 28(24):3240–3247, 2012.
5. A Biegert and J Sding. Sequence context-specific profiles for homology searching. *Proceedings of the National Academy of Sciences of the United States of America*, 106:3770–3775, March 2009.
6. Grzegorz M Boratyn, Alejandro A Schffer, Richa Agarwala, Stephen F Altschul, David J Lipman, and Thomas L Madden. Domain enhanced lookup time accelerated blast. *Biology direct*, 7:12, April 2012.
7. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
8. Xiuquan Du, Shiwei Sun, Changlin Hu, Yu Yao, Yuanting Yan, and Yanping Zhang. Deepppi: Boosting prediction of protein-protein interactions with deep neural networks. *Journal of chemical information and modeling*, 57:1499–1510, June 2017.
9. Tanlin Sun, Bo Zhou, Luhua Lai, and Jianfeng Pei. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC bioinformatics*, 18(1):277, 2017.
10. Matt Spencer, Jesse Eickholt, and Jianlin Cheng. A deep learning network approach to ab initio protein secondary structure prediction. *IEEE/ACM transactions on computational biology and bioinformatics*, 12:103–112, 2015.
11. Sheng Wang, Jian Peng, Jianzhu Ma, and Jinbo Xu. Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6:18962, January 2016.
12. Pietro Di Lena, Ken Nagata, and Pierre Baldi. Deep architectures for protein contact map prediction. *Bioinformatics (Oxford, England)*, 28:2449–2457, October 2012.
13. Rhys Heffernan, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Capturing non-local interactions by long short term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers, and solvent accessibility. *Bioinformatics (Oxford, England)*, April 2017.
14. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
15. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
16. Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
17. Jack Hanson, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. *Bioinformatics*, 33(5):685–692, 2016.
18. Louis Kim, Jacob Harer, Akshay Rangamani, James Moran, Philip D Parks, Alik Widge, Emad Eskandar, Darin Dougherty, and Sang Peter Chin. Predicting local field potentials with recurrent neural networks. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 808–811. IEEE, 2016.
19. Robert D Finn, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Jaina Mistry, Alex L Mitchell, Simon C Potter, Marco Punta, Matloob Qureshi, Amaia Sangrador-Vegas, et al. The pfam protein families database: towards a more sustainable future. *Nucleic acids research*, 44(D1):D279–D285, 2016.
20. Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 2017.
21. Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.
22. Julian Gough, Kevin Karplus, Richard Hughey, and Cyrus Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *Journal of molecular biology*, 313(4):903–919, 2001.
23. Michael Gribskov and Nina L Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. *Computers & chemistry*, 20(1):25–33, 1996.
24. Philip Bradley and David Baker. Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. *Proteins: Structure, Function, and Bioinformatics*, 65(4):922–929, 2006.
25. Jianlin Cheng and Pierre Baldi. Three-stage prediction of protein β-sheets by neural networks, alignments and graph algorithms. *Bioinformatics*, 21(suppl_1):i75–i84, 2005.