Bioinformatics, 34(14), 2018, 2487–2489
doi: 10.1093/bioinformatics/bty116
Advance Access Publication Date: 27 February 2018
Applications Note

Sequence analysis

SECLAF: a webserver and deep neural network design tool for hierarchical biological sequence classification

Balázs Szalkai[1,*] and Vince Grolmusz[1,2,*]

[1]PIT Bioinformatics Group, Institute of Mathematics, Eötvös University, H-1117 Budapest, Hungary and [2]Uratim Ltd, H-1118 Budapest, Hungary

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on August 14, 2017; revised on February 22, 2018; editorial decision on February 24, 2018; accepted on February 26, 2018

Abstract

Summary: Artificial intelligence tools are gaining more and more ground each year in bioinformatics. Learning algorithms can be taught for specific tasks by using the existing enormous biological databases, and the resulting models can be used for the high-quality classification of novel, un-categorized data in numerous areas, including biological sequence analysis. Here, we introduce SECLAF, a webserver that uses deep neural networks for hierarchical biological sequence classification. By applying SECLAF for residue-sequences, we have reported [Methods (2018), https://doi.org/10.1016/j.ymeth.2017.06.034] the most accurate multi-label protein classifier to date (UniProt—into 698 classes—AUC 99.99%; Gene Ontology—into 983 classes—AUC 99.45%). Our framework SECLAF can be applied for other sequence classification tasks, as we describe in the present contribution.

Availability and implementation: The program SECLAF is implemented in Python, and is available for download, with example datasets at the website https://pitgroup.org/seclaf/. For Gene Ontology and UniProt based classifications a webserver is also available at the address above.

Contact: grolmusz@pitgroup.org or szalkai@pitgroup.org

1 Introduction and motivation

New biological sequences are identified and submitted to public repositories by the thousands every day. The classification and annotation of these data is a demanding task. A frequent requirement is the hierarchical classification of the data, where data items need to be inserted in a pre-defined hierarchy, like a phylogenetic tree, a protein ontology- or a gene ontology graph. One possible solution for sequence classification could be the application of advanced artificial intelligence tools, such as artificial neural networks (McCulloch and Pitts, 1943). In a previous work (Szalkai and Grolmusz, 2018) we have constructed a framework, called SECLAF (Sequence Classification Framework), and have demonstrated its considerable power by multi-label classification of UniProt (UniProt Consortium, 2009) and Gene Ontology (Gene Ontology Consortium, 2015) entries. As we have demonstrated in (Szalkai and Grolmusz, 2018), the SECLAF produces the most accurate artificial neural network for residue sequence classification to date (for UniProt–into 698 classes–AUC 99.99%; for Gene Ontology–into 983 classes–AUC is 99.45%).

Here, we publish the downloadable SECLAF program, and, additionally, a pre-configured webserver at https://pitgroup.org/seclaf/. Our goal was to create a tool for designing deep neural networks which classify biological sequences. To make SECLAF user-friendly, only the input dataset (training and testing data) should be given in a certain format, but the neural network architecture and hyperparameters can be supplied in a human-readable JSON file. Preparation of the input data must be done by the user, but after that, no more coding is required.

2 Materials and methods

We implemented SECLAF in Python 3, using the neural network library Tensorflow (Abadi *et al.*, 2016; Abadi, 2016; Rampasek and

Goldenberg, 2016). Tensorflow is a relatively new framework created by Google which allows one to define and train neural networks at various levels of abstractions. We chose Tensorflow because it is easy to install, supports low level operations which eliminates the need for writing CUDA code when defining new layers and performs automatic differentiation. In addition, it is sufficiently fast when compared to the other options.

3 Implementation and usage

When training a neural network, the input of SECLAF should consist of the following files:

- the tree file: a hierarchy of the sequence classes (`classes.tre`),
- the training set: a file containing the training sequences along with their classification (`train_set.ann`),
- the test set: a file with the testing sequences and their annotations (classifications), which must be a file with the same structure as `train_set.ann`, and
- a file containing the network configuration and various necessary parameters for training, testing and inference (`config.json`).

SECLAF can perform a hierarchical classification of sequences. This includes non-hierarchical classification as a special case: if the classes are pairwise disjoint and there is no implication between class membership, the class hierarchy file should only contain a list of all classes with no parent classes specified, along with the textual descriptions of the classes. However, if there is an implication relationship between some or all of the classes, e.g. they can be organized into a tree with superclasses and subclasses, then this file should also contain the logical implications between class memberships. For example, if all sequences in class A also belong to class B, then being a member of class A logically implies being a member of class B. In other words, a relation can be defined on the classes, which we will call `is_a` after Gene Ontology terminology (Gene Ontology Consortium, 2015). Then `A is_a B` would mean that all the sequences in class A are members of class B as well. In this case, the line describing class A in the tree file must also contain a list of all the classes that are implied by A, i.e. those classes X for which `A is_a X`.

SECLAF will use the information about class hierarchy in both training and inference. In the training and test sets, the superclasses do not have to be present if the sequences are properly annotated with their corresponding subclasses because SECLAF will autocomplete the annotations by including all parent classes and their parents. In addition, when doing inference, SECLAF will output a subgraph of the class hierarchy with no outgoing edges, meaning that if class A is an output for some sequence S and another class B is (indirectly or directly) implied by A, then B will also be present in the output corresponding to sequence S. The exact format required for the class hierarchy file (and also for the sequence container files) is described in the readme file of SECLAF.

SECLAF implements a multi-label binary cross-entropy classification loss on the output neurons (each of which represents a possible label), specified in detail in (Szalkai and Grolmusz, 2018). There are a few minor differences because of the class hierarchy. When training, all annotations (label sets for sequences) are augmented with the possible ancestors of the labels in tree. This is to ensure that the annotations in the training set are consistent, i.e. if the network is trained to classify a sequence into a specific class, then it is also trained to classify that sequence into the parent class (and all possible ancestor classes). When testing, inclusion of ancestor nodes is not enforced, this should be learned by the network itself and thus the user can verify whether their trained network can produce consistent labellings of sequences or not. Inconsistencies in hierarchical predictions may only arise when a node is predicted but one or more of its parents are not. In this case, the predictions may be augmented by including all parents of all nodes that were predicted.

In the configuration file, one can configure basic neural network hyperparameters such as the learning rate, the learning rate decay schedule, the weight decay, the batch size, the number of iterations and parameters concerning class balancing. Constraints on the input sequences and classes can also be given: their minimum length (the neural network will have a lower bound on sequence length depending on the architecture), maximum length (if overly long sequences cannot fit into GPU memory), the minimum class size (number of sequences) and the maximum depth in the class hierarchy to consider.

The input sequence encoding must also be specified in the configuration file, as the neural network cannot accept character sequences, only numeric values. SECLAF can encode both DNA and protein sequences, but they cannot be mixed. Multiple encoding methods are available. The most simple one (SimpleDnaEncoder, SimpleAminoAcidEncoder) assigns the elements of a 4- or 20-dimensional standard basis (i.e. one-hot vectors) to each nucleotide or amino acid. A compact encoding method is available for DNA sequences (CompactDnaEncoder), which assigns a 3-dimensional vector to each nucleotide: the three components are all binary and correspond to the purine/pyrimidine, strong/weak and amino/keto dichotomies. Another method (CompactAminoAcidEncoder) assigns a 6-dimensional vector to each amino acid based on its chemical properties, and the last one (BigAminoAcidEncoder) assigns the concatenation of the two kinds of vectors (20- and 6-dimensional) to each amino acid, thus yielding a 26-dimensional vector. For example, BigAminoAcidEncoder will assign a matrix of size $L \times 26$ to an input sequence with length L. If N denotes the minibatch size, then a whole minibatch of sequences will be assigned a 3-rank array with shape $N \times L \times 26$. The network architecture must also be configured in the `config.json` file. The architecture should be given as a list of layers, excluding the input layer. The last one in the list will be the output layer, which must be a fully connected layer with the same number of outputs as the number of classes selected for classification. SECLAF supports the following layers: 1-dimensional convolution, 1-dimensional max pooling, batch normalization, dropout, global max-pooling and fully connected (dense). As the input of the network has a variable length, while its output has a fixed length, SECLAF requires a global max-pooling layer at a point, after which only batch normalization, dropout and fully connected layers are allowed.

Two examples are available with SECLAF to demonstrate how to use the program. One example classifies proteins into 983 Gene Ontology classes; the other one is the same network architecture applied for protein classification into 698 UniProt families. These networks use all the supported layer types, so they provide a comprehensive example for describing a neural network in SECLAF. Pre-trained networks are available for download for both examples at https://pitgroup.org/static/seclaf_pretrain.

We remark that the running time of the training phase in Example 1, where the number of sequences in the training set was 521 527, for 150 000 iterations, was 28 h on a single Geforce GTX 750Ti GPU with 4GB RAM. The training on a single Intel(R) Core(TM) i7 860 CPU at 2.80 GHz would have taken 209 h (8.7 days). This means that, according to our measurements, training is about 7.4 times slower on this particular CPU that this particular GPU. The gap is expected to be larger for high-end GPUs like the Titan X.

Funding

This work was supported through the new national excellence program of the Ministry of Human Capacities of Hungary [to B.S.]; and in part by the VEKOP-2.3.2-16-2017-00014 program and the NKFI-126472 grant of the Ministry for National Economy and the National Research, Development and Innovation Office of Hungary, respectively [to V.G.].

Conflict of Interest: none declared.

References

Abadi,M. (2016) Tensorflow: learning functions at scale. In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, Nara, Japan, 2016. pp. 1–1. ACM, New York, NY, USA.

Abadi,M. *et al*. (2016) Tensorflow: a system for large-scale machine learning. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. Savannah, GA, USA, USENIX Association, pp. 265–283. https://www.usenix.org/conference/osdi16

Gene Ontology Consortium. (2015) Gene ontology consortium: going forward. *Nucleic Acids Res*., **43**, D1049–D1056.

McCulloch,W.S. and Pitts,W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys*., **5**, 115–133.

Rampasek,L. and Goldenberg,A. (2016) TensorFlow: biology's gateway to deep learning? *Cell Systems*, **2**, 12–14.

Szalkai,B. and Grolmusz,V. (2018) Near perfect protein multi-label classification with deep neural networks. *Methods*, **132**, 50–56.

UniProt Consortium. (2009) The universal protein resource (UniProt) 2009. *Nucleic Acids Res*., **37** (Database issue), D169–D174.