Sequence analysis

Sequence-based protein function prediction using convolutional neural networks

Jeremy Kim

Department of Computer Science, University of Central Florida, Orlando, FL 32825, United States.

Abstract

Motivation: Recent developments in genome sequencing have resulted in massive volumes of protein sequence data. We must first link each protein sequence to relevant biological processes, cellular components, or molecular functions in order to make use of such data. This is referred to as the GO term annotation prediction problem. In other words, the difficulty is determining how to predict GO term annotations from protein sequence data. Data labeling by hand takes a long time, thus we need to automate this procedure.
Results: Recently, Zuallaert *et al.* (2019), studied the effects of different encoding strategies on protein function prediction performance using a Convolutional Neural Network (CNN). I reimplemented a condensed version of their work, using only one-hot encoded and ad-hoc trainable embeddings encoding methods and omitting trigrams. Despite using identical parameters, I could not reproduce the same results but observed a similar trend of simpler encodings doing better. Next, to improve the results, I supplemented the unigram one-hot encoded input with six chemical properties of amino acids and obtained slightly better results, particularly for the BP and MF functions, which may be due to BP and MF having a stronger correlation to chemical properties of amino acids than CC.
Contact: jihyungkim@knights.ucf.edu

1 Introduction

Protein function prediction has many important applications in understanding diseases and drug target discovery. Despite its importance, the process of annotating proteins with their function is very labor-intensive and time-consuming, therefore, finding an automated solution will accelerate developments in this area.

Various methods have been proposed for predicting protein functions from their sequence data. One of the first methods that has been used successfully for prediction protein functions is based on sequence similarity which mainly uses BLAST. Hennig *et al.* (2003) developed the GOblet software that uses BLAST to search a database of proteins and assign the Gene Ontology (GO) terms of similar sequences to the unknown sequence. More recently deep learning has been successfully applied to protein function prediction. Kulmanov *et al.* (2018b) used ad-hoc trainable embeddings of trigrams as input for a Convolutional Neural Network (CNN). It also incorporates hierarchical GO information to improve the results. Zuallaert *et al.* (2019) used different encoding strategies for the input to a CNN and studied the biological significance of these encodings and its effects on prediction results.

Here, I reimplement unigram and bigram one-hot encoding and ad-hoc trainable embedding encoding schemes from (Zuallaert *et al.*, 2019) in conjunction with the same CNN model used by them. In order to improve the results of the implementation, I experimented with approach, augmenting chemical properties of the amino acids with the unigram one-hot encoding.

2 Methods

2.1 Dataset

I used the same dataset that was used by Zuallaert *et al.* (2019) and Kulmanov *et al.* (2018b) and can be accessed from (Kulmanov *et al.*, 2018a). Protein sequences from SwissProt dataset were filtered based on their annotation's experimental evidence code (EXP, IDA, IPI, IMP, IGI, IEP, TAS and IC), maximum sequence length (1002), and not having ambiguous amino acid codes (B, O, J, U, X, Z). Furthermore, a minimum number of annotations were required for each GO class (50 for MF or CC and 250 for BP). The final dataset has 589 MF classes with 25224 training samples and 6306 testing samples, 932 BP classes with 36380 training samples and 9096 testing samples, and 439 CC classes with 35546 training samples and 8887 testing samples

2.2 Input representation

Following (Zuallaert *et al.*, 2019) the input to the model is the n-gram of the sequence of amino acid (AA) codes, that then is encoded using one of the following approaches:

- **One-hot encoding:** for each term of the n-gram, assign a vector consisting of all zeros, except for a one at the position reserved for that term.
- **Ad-hoc trainable embedding:** initializes a random vector of size v for each term of the n-gram, and v is learned along with other network parameters during training

Table 1. Original CNN model structure. Output shape is for MF model with unigram one-hot encoding

Layer	Output Shape
Encoding	(1002, 21)
Conv + Relu (64 filters of size 9, dropout = 0.2)	(994, 64)
Max Pooling (size 3, stride 3)	(331, 64)
Conv + Relu (64 filters of size 7, dropout = 0.2)	(325, 64)
Max Pooling (size 3, stride 3)	(108, 64)
Conv + Relu (64 filters of size 7, dropout = 0.2)	(102, 64)
Max Pooling (size 3, stride 3)	(34, 64)
K Max Pooling (K = 10)	(10, 64)
Flatten	(640)
Fully Connected (32 units)	(32)
Sigmoid Output (One for each GO class)	(589)

2.3 Original comparison method

The model developed by (Zuallaert *et al.*, 2019) compares the results. It is a CNN with one encoding layer, followed by three layers of convolution, Rectified Linear Unit (ReLU), drop out, and max pooling. Following that is a dynamic K-max pooling layer, which is utilized to provide a fixed output size independent of input size by picking the k largest values in each channel while maintaining their original order. Finally, for each GO class, there is a fully connected layer with a sigmoid output. The models were trained using a binary cross entropy loss and an Adam optimizer with a learning rate of 0.001, a batch size of 64, and 15 epochs. 87.5% of the dataset was used for training, with the remaining 12.5% used for validation. Table 1 shows the original model structure with parameters taken from (Zuallaert *et al.*, 2019). It also shows the output dimensions for a model trained with MF data that uses the unigram one-hot encoding.

2.4 Amino acid chemical properties augmentation method

Because a protein's function is determined by the chemical characteristics of its amino acids, enhancing the sequence data with these qualities should yield more meaningful information that may be employed in the function prediction task. The chemical parameters listed below were chosen. (Szalkai and Grolmusz, 2018):

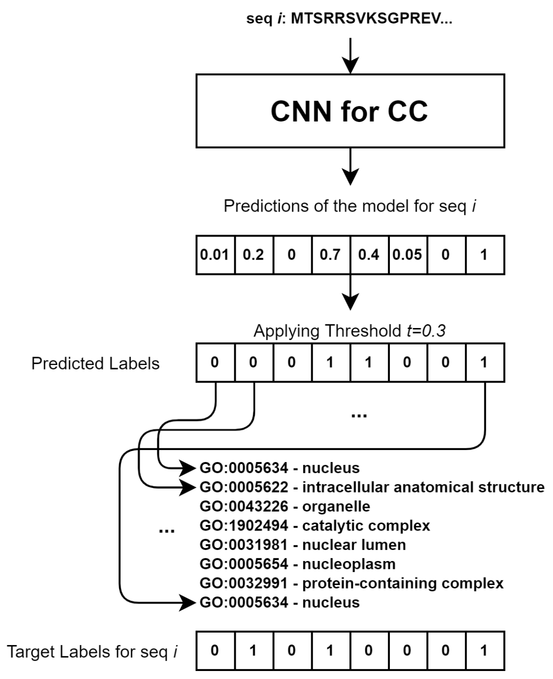
1. Expected charge: (1: positive, -1: negative, 0: neutral)
2. Hydrophobicity: from -4.5 to 4.5
3. Polar: yes or no (1 or 0)
4. Aromatic compound: yes or no (1 or 0)
5. Has a hydroxyl group: yes or no (1 or 0)
6. Has a sulfur atom: yes or no (1 or 0)

These are easily obtained by combining the amino acid letters. The one-hot encoded unigram (20 dimension) is supplemented with these 6 integers, resulting in a vector of 26 dimensions (27 dimensions if the null data one-hot encoding is included) for each amino acid in the sequence. Except for the input, which is reinforced with chemical properties of the amino acids, the model employed here is the same as the initial model with the same parameters displayed in Table 1.

2.5 Evaluation

Evaluation metrics are based on metrics defined by Critical Assessment of protein Function Annotation (CAFA) challenges (Jiang *et al.*, 2016) and is computed as follows:

$$p(t) = \frac{1}{m(t)} \sum_{i=1}^{m(t)} \frac{f_i(t)}{g_i(t)},$$



seq *i*: MTSRRSVKSGPREV...
CNN for CC
Predictions of the model for seq *i*
0.01 0.2 0 0.7 0.4 0.05 0 1
Applying Threshold *t=0.3*
Predicted Labels
0 0 0 1 1 0 0 1
...
GO:0005634 - nucleus
GO:0005622 - intracellular anatomical structure
GO:0043226 - organelle
GO:1902494 - catalytic complex
GO:0031981 - nuclear lumen
GO:0005654 - nucleoplasm
GO:0032991 - protein-containing complex
GO:0005634 - nucleus
...
Target Labels for seq *i*
0 1 0 1 0 0 0 1

Fig. 1. Evaluation process for a sample sequence for CC data

$$r(t) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_i(t)}{h_i},$$

$$F_{max} = \max_t \left\{ \frac{2p(t)r(t)}{p(t)+r(t)} \right\},$$

where $t \in (0, 1)$ is the threshold that converts the sigmoid output to 0 or 1 as predicted label, $p(t)$ is the average precision for a given t, $m(t)$ is the number of sequences with at least one prediction above threshold t, $f_i(t)$ is number of correct predictions for a given t for sequence i, $g_i(t)$ is the number of predictions for a given t for sequence i, $r(t)$ is the average recall for a given t, n is the total number of sequences, h_i is the number of target labels for sequence i, and F_{max} is the maximum F-score obtained by varying t from 0 to 1.

Figure 1 shows the process of evaluation for a sample sequence. After applying the threshold to the sigmoid output of the CNN, we get a vector of zeros and ones, where the presence of a 1 in a particular position indicates that the sequence belongs to that class. Comparing the predicted labels with the target labels (ground truth) allows us to calculate the F_{max} when the process is repeated for all sequences and thresholds. Since we have infinite values between 0 and 1, in practice the threshold is incremented by 0.01 from 0 to 1, and the F-score is calculated for that t. The final F_{max} is calculated as the maximum of all calculated F-scores.

3 Results

The studies were carried out on a Windows 10 laptop equipped with an Intel Core i7-4720HQ CPU running at 2.6 GHz, 16 GB of RAM, and an Nvidia GeForce GTX 970M graphics card. The models were written in Python 3.7 with Tensorflow 2.1. A distinct model for BP, MF, and CC is trained for each of the models. Each result is provided as the average of ten runs, with the associated 95% confidence intervals. The findings of the proposed models will be presented in the sections that follow.

3.1 Original model implementation

I used the same model and parameters as the original to run the experiments. Figure 2 shows the comparison of the results reported by (Zuallaert *et al.*,

Table 2. Comparison between results of original from (Zuallaert et al., 2019) with my original, my original with chemical properties augmentation method

Function	Original	My Original	My Original with chemical properties augmentation method
BP	0.358	0.310 (+/- 0.00458)	0.317 (+/- 0.00137)
MF	0.411	0.383 (+/- 0.00515)	0.393 (+/- 0.00376)
CC	0.595	0.582 (+/- 0.00335)	0.589 (+/- 0.00203)

2019) on the left side and the results of my implementation of the original model on the right side which also include the 95% confidence intervals shown by the error bars.

I was not able to get the same results as (Zuallaert *et al.*, 2019) which can be due to some implementation errors on my part or some inaccuracies in the reported parameters by (Zuallaert *et al.*, 2019), but without having access to their code, it is difficult to find the exact cause of this discrepancy.

Although I was unable to get the same results but there are some general trends that are still manifested in my results similar to (Zuallaert *et al.*, 2019). The first one is the fact that increasing n for n-grams reduces the F_{max}, which is not very surprising since increasing n, results in more sparse input which makes the task of finding efficient representation more difficult. The second trend is that for unigram ad-hoc trainable embedding, increasing the vector size v, had no impact on F_{max}. This indicates that the unigram can be adequately represented by a vector of size 5 and increasing v results in no improvements. The last trend is that for bigrams increasing v results in some small improvement up to a point but stops shortly after. This is because the size of the bigrams vocabulary is much larger than unigrams which require larger v, but because of the redundancy present in bigram representation, continuing to increase v will not yield better results.

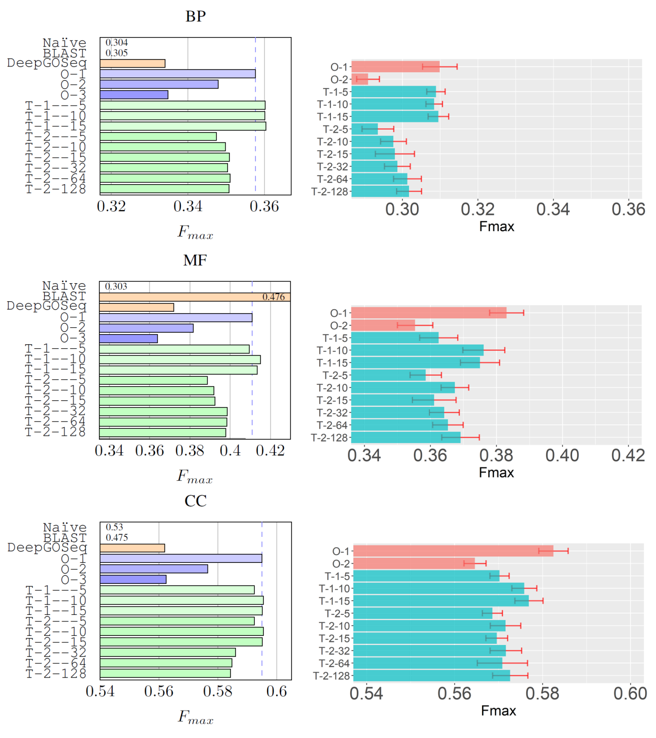


Fig. 2. Comparison between original model from (Zuallaert et al., 2019) on the left side and my implementation on the right side. The error bars on the right side show the 95% confidence intervals.

Note. Left-hand side graphs are from Fig. 5 from (Zuallaert et al., 2019)

3.2 Amino acid chemical properties augmentation results

The results of augmenting the input to the original model with chemical properties of the amino acids are shown in Table 2, which also compares the results to the original model results for the unigram one-hot encoding from (Zuallaert *et al.*, 2019) and my implementation of the original model for the unigram one-hot encoding.

Although it still didn't manage to reproduce the original model's results, it improved the results of my implementation of the original model by 2.3% for BP, 2.6% for MF, and 1.2% for CC. It's noticeable that the largest performance increases were for BP and MF, which might be due to a slightly stronger correlation between biological processes (BP) and molecular functions (MF) with chemical properties of amino acids compared to cellular components (CC). The effect is small because the chemical properties that I used were easily deducible from the sequence such that a CNN model can implicitly learn them automatically, but this might be a hint that using other chemical properties that are more complex might have a greater impact on the performance since the model have immediate access to these useful but hard to extract features and does not need to learn them from scratch.

4 Conclusion

Given the importance of protein function prediction in many areas of bioinformatics, it is very important to have automated systems that can accurately and efficiently predict the protein function's from their sequence data.

Deep learning methods have shown great success in solving difficult prediction tasks. A recent example of employing deep learning for solving the protein function prediction is the work done by (Zuallaert *et al.*, 2019). I implemented my original model based on their work and experimented with different models in order to improve their results. I first attempted to reproduce their implementation exactly but did not succeed in achieving the same results. Although I found the same trend as the original results in that using larger n for n-grams lowered the F_{max} score and ad-hoc trainable embedding didn't offer better results than one-hot encoding. Larger n-grams are more sparse and result in more network parameters that need to be learned and thus lower the model's performance. The added complexity of adding an additional encoding layer for ad-hoc trainable embedding was shown to not result in a better performance which could be due to having more parameters that need to be trained compared to a simple unigram one-hot encoding.

In my second model, adding chemical properties of amino acids to the input, showed improvement over my original model for BP and MF, but not for CC. This difference in protein function performance could be due to a more correlation between the chemical properties of amino acids and BP and MF compared to CC. This model was successful in getting better results than my original model's results.

References

Hennig, S., Groth, D., and Lehrach, H. (2003). Automated gene ontology annotation for anonymous sequence data. *Nucleic Acids Research*, **31**(13), 3712–3715.

Jiang, Y., Oron, T. R., Clark, W. T., Bankapur, A. R., D'Andrea, D., Lepore, R., Funk, C. S., Kahanda, I., Verspoor, K. M., Ben-Hur, A., *et al.* (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome biology*, **17**(1), 1–19.

Kulmanov, M., Khan, M. A., and Hoehndorf, R. (2018a). Dataset for deepgo. http://deepgo.bio2vec.net/data/deepgo/data.tar.gz. Accessed: 2021-02-11.

Kulmanov, M., Khan, M. A., and Hoehndorf, R. (2018b). Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, **34**(4), 660–668.

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

Zuallaert, J., Pan, X., Saeys, Y., Wang, X., and De Neve, W. (2019). Investigating the biological relevance in trained embedding representations of protein sequences. In *Workshop on Computational Biology at the 36th International Conference on Machine Learning (ICML 2019)*.