Technical University of Denmark

The language of life

The unemployed cells:
Marianne Helenius (s143829)
Hannah-Marie Martiny (s143830)

November 1, 2018
Course: 02456 Deep learning
Course responsible: Ole Winther
Supervisors: José Armenteros and Alexander Rosenberg Johansen

DTU Bioinformatics
Department of Bio and Health Informatics

Contents

Abbreviations

ANN	Artificial Neural Network
CBOW	Continuous bag-of-words
CNN	Convolutional Neural Network
GloVe	Global Vectors
NLP	Natural Language Processing
PTB	The Penn Treebank

1 Introduction

Deep learning methods has previously been applied in fields such as image and text recognition, but here we propose to also apply it to the language of life. Language of life essentially refers all of the information that is encoded by our genetic sequences, and we propose to view these sequences as the sentences that makes up the vocabulary of life.

The language of life is built by protein sequences, and the biological alphabet consists of 21 amino acids denoted by the letters seen in Table 1. This project aims to predict an amino acid y at a position t, $p(y_t)$, where the input can be the amino acid distribution or a partial sequence either given k places before $p(y_t|y_{t-k})$, after $p(y_t|y_{t+k})$ or both $p(y_t|y_{t-k}, y_{t+k})$. This will be achieved by using n-grams and convolutional neural networks.

Table 1: The 21 amino acids.

Amino acid	**One-letter code**	**Three-letter code**
Alanine	A	Ala
Cysteine	C	Cys
Aspartic acid	D	Asp
Glutamic acid	E	Glu
Phenylalanine	F	Phe
Glycine	G	Gly
Histidine	H	His
Isoleucine	I	Ile
Lysine	K	Lys
Leucine	L	Leu
Methionine	M	Met
Asparagine	N	Asn
Pyrrolysine	O	Pyl
Proline	P	Pro
Glutamine	Q	Gln
Arginine	R	Arg
Serine	S	Ser
Threonine	T	Thr
Valine	V	Val
Tryptophan	W	Trp
Tyrosine	Y	Tyr

1.1 Language models

Natural language processing (NLP) is the area of deep learning that aims to analyze and predict sentences. Previous language models have been built by training a neural network on a vocabulary, i.e. a large sample of words in order to predict e.g. the context or related words in a sequence. We will apply some of these concepts to our purpose in order to process the natural language of protein sequences, as our sequences are essentially just

sentences. NLP models have already been applied to protein sequences in order to predict protein families or structures. [1]

N-grams are a method for representing text strings as slices of text with length n [2, 10]. This method of representation greatly increase the existing vocabulary, but also enables the model to detect which strings occur together.

If you were to represent the protein sequence "M A K L E P V V" as n-grams, it would be the following:

- *Unigrams ($n = 1$):* "M", "A", "K", "L", "E", "P", "V" and "V".
- *Bigrams ($n = 2$):* "MA", "AK", "KL", "LE", "EP", "PV" and "VV"
- *Trigrams ($n = 3$):* "MAK", "AKL", "KLE", "LEP", "EPV" and "PVV".

1.2 Convolutional neural networks

Convolutional neural networks (CNNs) are not only used in image recognition but can also be used to solve NLP tasks. CNNs takes advantage of having layers with convolving filters by applying them to local features in the input [4]. Features can be extracted independently by their position in the sentence if the filters are convolved with the n-gram at every position. Following this is a dynamic pooling layer and a non-linearity and all of this makes a feature map [3].

1.3 Related work

Language modelling has expanded in recent years with many expansions and modifications to existing models. The use of protein sequences in terms of language modelling has only just begun and there is still much more to develop in this area.

Some of the more popular models which have helped develop the field are described below. The methods of these models will also be taken into account, when building the protein language models in this project.

Word2Vec are a series of language models, which attempt to find appropriate vector representations or embeddings of words in a vocabulary. The structure of the models used to find these representations may vary, but essentially artificial neural networks (ANNs) are trained on a vocabulary, where the matrix between the input and hidden layer will represent the words. [6, 7]

These models will use a word and the words in its proximity, where the proximity is defined by a distance, k. Two examples of this type of model are seen below. These two models both do not consider the order of the words.

- *Continuous bag-of-words (CBOW)*: This model predicts the middle word from the surrounding words. [6, 7]

- *Continuous skip-gram:* From the current word, this model is able to predict words in proximity. [7]

The output of these models will be an M length vector, where M is the number of words, containing the probability of each word in the vocabulary being the middle word or in proximity of the middle word. [6, 7]

Global Vectors (GloVe) is another model to make word representations. This method uses the statistics of word occurrences, which is represented by a word co-occurrence matrix. [9]

2 Methods

The code for this project is freely available at `https://github.com/mari756h/The_unemployed_cells`. All code is written in Python 3, using the package Pytorch for the neural network models[1]. The code is executed using CPU/GPU.

2.1 Data description

In this project, the models are developed based on the Penn Treebank data, before applying the models to the protein sequence data set. This is done to verify that the design is working.

The Penn Treebank (PTB) is a collection of more than 4.5 million American English words, collected from 1989 to 1992. This collection is an annotated corpus, meaning that the words are viewed in structured sets of texts. [5]

The actual data for this project is protein sequences of varying lengths, where the 21 amino acids are denoted by different letters of the alphabet. Here, the training data consists of 126,795 protein sequences, the test data holds 39,698 sequences, and the validation set consists of 19,852 protein sequences. The sequences are held in .txt files, where each row corresponds to a sequence and the amino acids are white-space separated. An example of these sequences can be seen in Listing 1.

Listing 1: Example of amino acid sequences. *This is an example of how the sequences are stored, however, these are not actual sequences from the data.*

```
1 M E E A K V E A K D G T I S V A S A F S G H Q Q A V H D S D H K F
    L T Q A V E E A Y K G V D
2 M P S A F E D S L R N S T S F R P Y C L L N R K F S S S R F W K P
    R Y S C V N L S I K D I L E P S A P E
```

[1] Pytorch documentation: `https://pytorch.org/`

2.2 Data and feature representation

The data will be represented as one-hot-encoded vectors of length M, where M is the number of words in the vocabulary, the number of different amino acids in the amino acid vocabulary. The size of the amino acid vocabulary is in its essence 21, but it can be expanded by representing the protein sequences as *n-grams*. An example of this is one-hot-encoding as seen in Table 2 for a few amino acids as 1-grams, meaning that each amino acid is one "word" in the vocabulary.

Table 2: One-hot-encoding of amino acid 1-grams.

Amino acid	**One-letter code**	**One-hot-encoding**
Alanine	A	$[1, 0, 0, ..., 0, 0, 0]$
Cysteine	C	$[0, 1, 0, ..., 0, 0, 0]$
Aspartic acid	D	$[0, 0, 1, ..., 0, 0, 0]$
...	...	...
Valine	V	$[0, 0, 0, ..., 1, 0, 0]$
Tryptophan	W	$[0, 0, 0, ..., 0, 1, 0]$
Tyrosine	Y	$[0, 0, 0, ..., 0, 0, 1]$

This representation will, however, assumes that each amino acid is independent, which is why learning an embedded representation in our models will help gain knowledge on their relationship. This knowledge can improve the overall model prediction.

2.3 Model architecture

The model will be of an n-gram CNN architecture. The input layer will be the protein sequences, and the first layer does one-hot-encoding to the input (see Table 2). The second layer does n-gram normalization to extract features from the input sequences.

Following this is a convolutional layer that has multiple filter widths and feature maps and then a pooling layer. A operation in the convolutional layer includes a filter w that can be applied to a window of h amino acids in order to create a new feature. A feature c_i can be generated from a window of amino acids $x_{i:i+h-1}$ by equation Equation 1.

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{1}$$

b is the bias term and f is a non-linear function. A feature map $c = \{c_1, c_2, ...c_{n-h+1}\}$ can be produced by applying the filter in Equation 1 to each possible window of amino acids.

Pooling is a way of capturing the most important features $\hat{c}$ of the feature map c, and can be done by taking the maximum value over the feature map as seen in Equation 2. [4]

$$\hat{c} = \max\{c\} \tag{2}$$

The output of the pooling layer will be processed by the softmax function (see Equation 3), so that a vector is outputted consisting of predicted probabilities of each amino acid being

the next in the sequence. [8]

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3}$$

In Equation 3, x_i is the weighted input and the denominator is the sum of all the output nodes. The softmax function ranges 0 to 1 and the outputted vector will sum to 1.

References

[1] Ehsaneddin Asgari and Mohammad R. K. Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PLOS ONE* 10.11 (Nov. 2015). Ed. by Firas H Kobeissy. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0141287. URL: https://dx.plos.org/10.1371/journal.pone.0141287.

[2] William Cavnar, William Cavnar, and John M. Trenkle. "N-Gram-Based Text Categorization". In: 1994, pp. 161–175. DOI: 10.1.1.21.3248.

[3] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A Convolutional Neural Network for Modelling Sentences Nal". In: *Proceedings ofthe 52nd Annual Meeting ofthe Association for Computational Linguistics*. 2014, pp. 655–665. DOI: http://dx.doi.org/10.4103/0377-4929.42513. arXiv: arXiv:1404.2188v1.

[4] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014), pp. 1746–1751.

[5] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. "Building a large annotated corpus of English: The Penn Treebank". In: *Computational Linguistics* 19.2 (1993), pp. 313–330. URL: https://dl.acm.org/citation.cfm?id=972475.

[6] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: (Oct. 2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[7] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: (Jan. 2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[8] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/index.html.

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha: Association for Computational Linguistics, 2014, pp. 1532–1543. URL: https://www.aclweb.org/anthology/D14-1162.

[10] Andrija Tomović, Predrag Janičić, and Vlado Kešelj. "n-Gram-based classification and unsupervised hierarchical clustering of genome sequences". In: *Computer Methods and Programs in Biomedicine* 81.2 (Feb. 2006), pp. 137–153. ISSN: 01692607. DOI: 10.1016/j.cmpb.2005.11.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169260705002361.