Machine Learning CS-433 Project II: Protein sequence matching

Andrea Giovanni Perozziello, Henrik Øberg Myhre, Jurriaan Marcus Hubertus Schuring
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—This research aims to develop a machine learning method to classify whether a given start and end amino acid sequence from a certain protein type belong together. For this purpose, homologs - proteins with the same function from different species - were used to train machine learning models. The protein sequences are split in two and hot-key encoded before we use different machine learning models to predict if two half sequences belong together. Logarithmic regression and neural networks both manage to classify this with high accuracy. However, the neural network proves to be better at finding patterns than the logistic regression, which allows for less preprocessing and hence a lower time complexity overall.**

I. Introduction

AlphaFold[1] provided ground-breaking insight into how protein chains are folded. It used the correlation of mutation rates at different amino acid positions within the same protein to see whether these mutations are independent or not. The mutation rates were calculated by comparing among homologs. If the mutations are dependent, then the amino acids are thought to be close in the three dimensional structure of the protein. AlphaFolds' findings are important because the structure of proteins determines their functionality.

This research aims to use this inherent information about the three-dimensional structure in the amino acid sequences. We aim to train a classifier to tell real and false combinations of start and end amino acid chains apart. Since amino acids in the start sequence matter for the end sequence and vice versa, there should be some sort of pattern that the classifier can learn.

The protein we used was the SIS 1 gene from the Saccharomyces cerevisiae (Baker's yeast). The protein has an important cellular function and therefore has many homologs across species (also in humans). True data were start and end sequences from the same protein, while false data was made by matching real start sequences with artificially created end sequences.

II. Data preprocessing

The raw data we used consisted of aligned amino acid sequences, so besides the 21 amino acids there were also "-" to encode point deletions or insertions. The total amount of homologs was 5 353 sequences, and the sequences were 352 amino acids long, see an example below.

```
MG--VDYYNVLQVDRNASDNDLKKAYRKLAMKWHPDKP-NNK
KKFKQISEAYEVLSDPQKKAVYDQYGLK---GN-PP-PDA--
G-----GG-P----------FNPRNADDIFAEFF-GF-PF-
------------------G------FSSSF-----G---G--
--A-GARKP----VTHDLRVSLEEIYSGCTKKMKISHKNPDG
SIEDKILTIEVKRGWKEGTKITFPKEGDQ-S--NNPADIVFV
LKDKPHNIFKRDGSDVIYPARISLREALCG-TVNVPTLDGR-
-TI--KDVIRPGMRRKVPGEGLPLPKTPEKRGDLIIEFEVIF
PERIPQTSKTVLEQVL
```

These sequences were saved to two different CSV-files consisting of:

1) All the real amino acid sequence combinations.
2) All possible false amino acid sequence combinations. We split the sequences into two

[1] "AlphaFold: a solution to a 50-year-old grand challenge in biology", https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

halves. The total number of possible artificially created sequences is

$$5\ 353 \times (5\ 353 - 1) = 28\ 649\ 256$$

since every first half is combined with every second half except for its actual continuation. The false file is randomly shuffled to randomize the extraction of false examples. The randomization is necessary because the raw data file was somewhat sorted.

We use these two files to create the final file for training and testing. Since our data contains many more false sequences than true, it is not suitable to include every false sequence. The final file is hence created by using a false-true ratio. All of the true data is always included, but the amount of false data vary according to the ratio. The start and end sequences are not combined because we want to keep them separate for different applications later.

III. Intermediate processing

The data that we feed the models is hot-key encoded. Due to limited storage and the space complexity of 22 characters hot-key encoding, we couldn't store all of this data before training the models. Therefore, we iterate through the data with a fixed batch size and perform all the remaining necessary steps during the training.

The first step is, as mentioned, hot-key encoding of the data. Each possible amino acid character in the start/end sequence is transformed to a hot-key encoded list, and all of the lists are added together. Then, the start and end sequences are separately reduced with PCA[2]. PCA (principal component analysis) is a method to reduce the dimensionality of the input while keeping most of the information in the data. We reduced the dimension from

$$22 \times 176 = 3\ 872$$

to 300, while still over 90% of variance was explained. The discrepancy between the one-hot encoded data and the reduced data is therefore fairly small and the amino acid sequences should be reasonably represented by the reduced vectors. Reducing the complexity will reduce the number of features provided as input to our models and will make it easier to compare the start and end sequence.

Lastly, we combined the start and end sequences by adding, multiplying or concatenating them together. The adding and multiplying should provide the models with data that shows the (non-)existence of correlations between the start and end sequences. For concatenation on the other hand, the model will have to learn how to combine the amino acids to correctly classify the data.

[2]"IncrementalPCA", https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html

IV. Models and optimization

We used two different machine learning methods and a total of three different permutations. Two of these were neural networks. The first permutation consisted of one hidden layer of size 50, 100 or 200. The other permutation consisted of two hidden layers of sizes 50, 100 or 200 and 20, respectively. The last method we used was logistic regression.

The data and all methods have been optimized by testing different parameters. The data preprocessing was optimized by testing different ratios of the true and false data. A ratio of 2 would mean that we include two fake sequences for every real sequence. The machine learning methods were optimized by testing different values for:

Parameters			
Sizes of hidden layers	50	100	200
Vector preprocessing	adding	multiplying	concatenating
True/false ratio	1	4	

TABLE I
Hyperparameters which were optimized All possible combinations of these parameters were used.

V. Results

All of our models performed better on the balanced data set, with the highest F score being 0.96, while the unbalanced data set gave an F score of 0.93. so we kept the true-false ratio at one. Hence, we used 5 353 real sequences and 5 353 fake sequences.

The logistic regression was able to well if the start and end vectors were multiplied (F1-score 0.85). When the sequences were added or concatenated

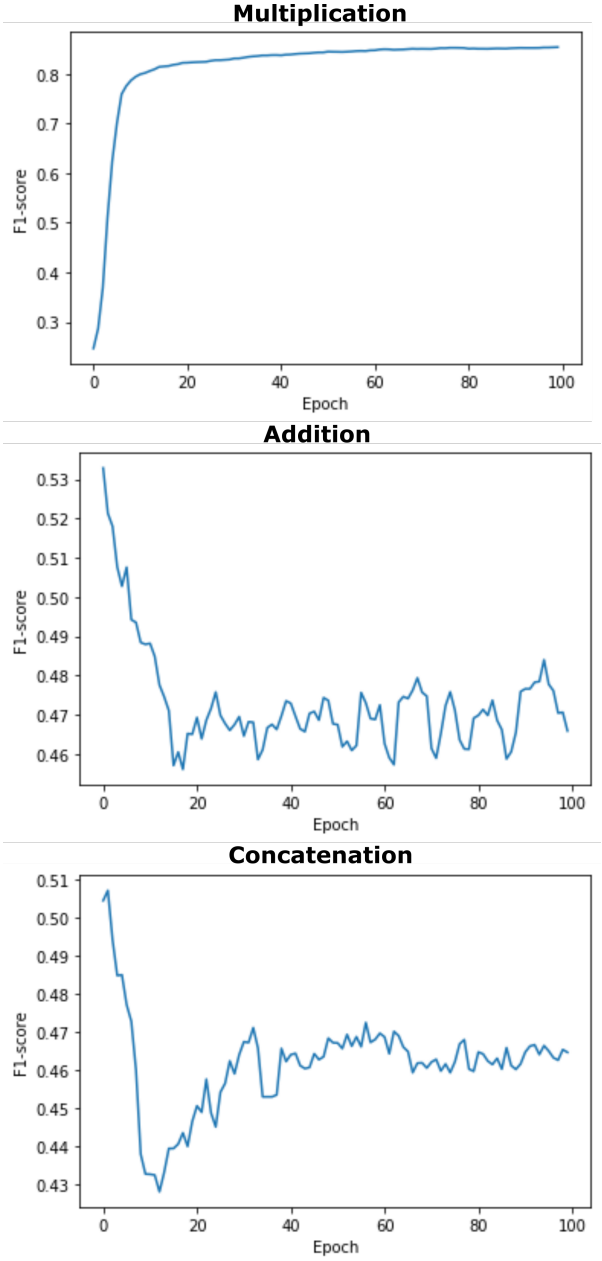

Fig. 1. **Logistic regression** The logistic regression model was able to learn when the sequences were multiplied, but not when they were added or concatenated. (lr = 0.001)

however, the logistic regression model did not learn at all (see figure 1).

The neural networks on the other hand could learn on the mutltiplied ($F1score = 0.94$), the added ($F1score = 0.96$) and the concatenated ($F1score = 0.96$) vectors. The scores are highly similar for one and two layered neural networks. We also see that the neural network with one hidden layer learned slower than the neural network with

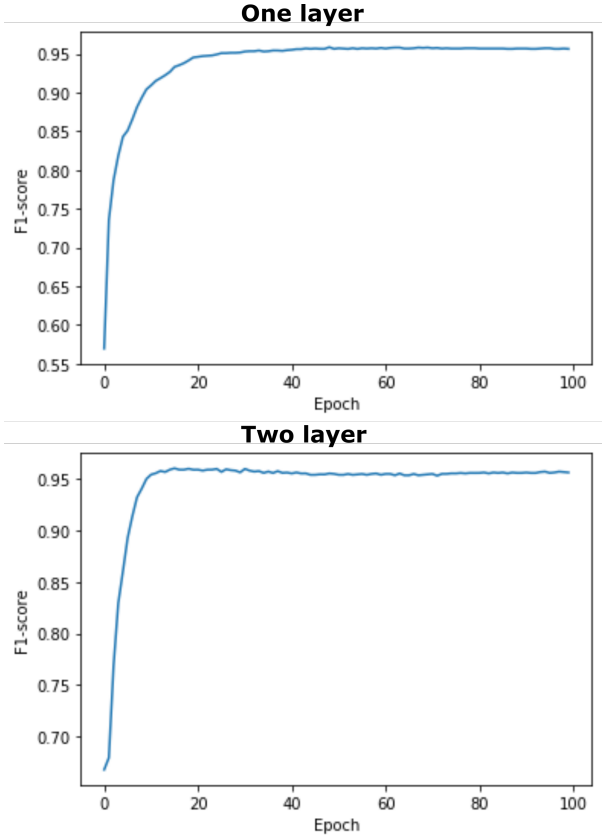

Fig. 2. **Comparison of neural networks** The progress of F1-scores for the best performing neural network configurations (see table II). The two layered neural network was slightly faster.

two hidden layers (see figure 2).

The optimal parameters and F1-scores per model were:

	logistic	1 layer NN	2 layer NN
Hidden layer size	-	50	200
Learning rate	0.001	0.001	0.001
Input method	Multiply	Concat.	Concat.
F1-score	0.85	0.96	0.96

TABLE II
OPTIMAL VALUES FOR THE HYPERPARAMETERS

VI. Conclusion

The best performing model was the neural network with one hidden layer. The other methods also performed well, but the logistic regression only learned if the data were by multiplication.

We have successfully classified real and fake sequences with high F1-scores at 0.96. In addition

to achieving a high accuracy, the training is fast (25 seconds for PCA and 1 second for the one layer neural net with 50 nodes for 30 epochs) due to the reduced dimensionality.

VII. Discussion

This research has highlighted a crucial difference between logistic regression and neural networks with one or more hidden layers. The logistic regression needed the hot-key encoded start and end sequences to be combined - by multiplying - to "see" the patterns. The reason why the neural networks did not need this is probably because of the hidden layer. The hidden layers learn to extract the information we got when we multiplied the start and end sequences. We can imagine that a node in the hidden layer combines two of the input features. The hidden layer in the neural net seems to learn which features it should combine to make a correct prediction. In the end it combines the features better than just multiplying the two vectors (what we did for logistic regression), since the neural networks get around 0.1 higher F1-scores than the logistic regression.

We also notice that the patterns the models find are simple enough for a single hidden layer neural network with just 50 nodes in the hidden layer. A double hidden layer does not perform better it only takes less epochs to maximize the learning. It is therefore redundant to use a bigger neural network.

To improve predictions, more false data could be included. To prevent false data from dominating and reducing the F1-score, the true data should be weighted more heavily in the training process. For this purpose weights could be added to the machine learning model. In this way, more training data will be available, resulting in higher accuracy and F1-score.

The ability to learn depends heavily on the entanglement of the proteins. More entangled proteins will lead to a higher accuracy than less entangled proteins. This is because there are more correlations between amino acids and hence more patterns to be found by the models. Since we have just tested for one specific protein type, we are not certain that the models will perform as good for other proteins.