Heinrich-Heine-Universität Düsseldorf
Computational Cell Biology
Alexander Kroll

Applications of Transformer Networks in Bio- and Cheminformatics

Worksheet 4

Submission Deadline:	**19. Mai 2024**, 11:59 pm
Discussion of solutions:	21. Mai 2024, 4:30 - 6:00 pm

Submission Instructions:

- Upload a Jupyter notebook with your solutions to the exercises that you solved on your own PC. Upload the Python scripts and log files for any code executed on the HPC.
- If you are submitting multiple files, zip them together.
- Submit your solutions by uploading the Jupyter notebook to https://uni-duesseldorf.sciebo.de/s/hCt1rTP23EeWmUC. The uploaded file should have the following filename: "lastname_studentID_worksheet4".

Excercise 4.1 *Gradient Boosting Hyperparameter Optimization* (50 Points)

For this task, you will need to familiarize yourself with the Python package hyperopt. You will perform a hyperparameter optimization for a gradient boosting model for predicting the optimal pH of proteins using the dataset from Worksheet 1. The input features will be the frequencies of all 1-mers and 2-mers, i.e. you will have input vectors of dimension 420. Implement the gradient boosting models using the Python package xgboost.

(a) Perform a hyperparameter optimization for a gradient boosting model (xgboost) using the hyperopt package. Search for the best set of hyperparameters among the following parameters with the following upper and lower bounds:

- Number of trees ("n_estimators"): Lower bound: 10, Upper bound: 500,
- Maximum tree depth: ("max_depth"): Lower bound: 1, Upper bound: 10,
- Learning rate: ("learning_rate"): Lower bound: 0.01, Upper bound: 0.5
- Min. child weight ("min_child_weight"): Lower bound: 1, Upper bound: 10,
- Lambda ("reg_lambda"): Lower bound: 0, Upper bound: 1,
- Alpha ("reg_alpha"): Lower bound: 0, Upper bound: 1.

Search for the best set of hyperparameters on the validation set by iterating over at least 200 different combinations of hyperparameters. Search for the set that leads to the lowest coefficient of determination R^2. If this takes too long on your PC, run the code on the HPC.

Hint: Consider that hyperopt tries to minimize a function but you want to maximize R^2.

(b) Select the set of hyperparameters that led to the best R^2 on the validation set in (a). Train a gradient boosting model with this set of hyperparameters on the training set and evaluate it on the test set, i.e., calculate MSE and R^2 for the test set.

Excercise 4.2 *ESM-2 embeddings for pH prediction* (50 Points)

(a) Compute ESM-2 (model: esm2_t6_8M_UR50D) protein embeddings for all sequences in the pH dataset. This should give you a single protein representation of dimension 320 for each protein sequence. For an example on how to compute these embeddings, see the Jupyter notebook *data/worksheet4/Get_ESM2_embeddings.ipynb*. If you cannot calculate the embeddings on your PC but instead need to use the HPC, you can find instructions in the same Jupyter notebook for how to move the ESM-2 model to the HPC and load it from a local file.

(b) Perform a hyperparameter optimization of a gradient boosting model as described in Exercise 4.1 (a) but with the ESM2-embeddings as input features.

(c) Train a gradient boosting model with this set of hyperparameters on the training set and validate it on the test set, i.e., calculate MSE and R^2 for the test set.

Excercise 4.3 *Fine-Tuning ESM-2* (70 Points)

In this exercise, we will fine-tune a ESM-2 model (model: esm2_t6_8M_UR50D) with 6 encoder layers for the task of predicting optimal pH. In the subparts of this exercise I will guide you through the necessary steps. I recommend training the model on the HPC if you cannot use a GPU on your own PC.

(a) When fine-tuning the ESM-2 model, we want to build a feed-forward network (for optimal pH prediction) on top of the updated classification token. By default, the ESM-2 model includes a classification token in its input sequence. For the following tasks, you need to find where it is placed in the tokenized sequence. The function *alphabet.to_dict()* might help you.

(b) Define a neural network that takes as input a protein amino acid sequence that is processed by the pre-trained ESM-2 model. After all input tokens are updated, the updated embedding of the classification token from layer 6 should be used as input for a fully connected neural network to predict the optimal pH of the protein.

(c) Define a function to divide your training and validation data sets into batches of batch size 8.

(d) Fine-tune the ESM-2 model for optimal pH prediction for at least 1 epoch on the training set. Set the loss function to mean squared error, the optimizer to the Adam optimizer, and the learning rate to 10^{-5}.

(e) Validate how well this fine-tuned ESM-2 model performs on the validation set for predicting the optimal pH: Calculate the MSE and the coefficient of determination R^2.

(f) Extract for all proteins (training, validation and test proteins) in the optimal pH data set the fine-tuned ESM-2 embeddings, i.e. the updated classification token embeddings from layer 6.

(g) Perform hyperparameter optimization for a gradient boosting model as described in Exercise 4.2 using the new and fine-tuned ESM-2 embeddings.

Note on using the HPC: If you plan to use a GPU on the HPC for this task, please first check if there are enough GPUs available. If you are logged into the HPC via the terminal, you can check this with the following two commands:

```
module load hpc-tools
gpus_free
```

Please use a GPU only if many GPUs are available, otherwise use a CPU. You can request a specific GPU, such as the RTX6000, by setting the following values in your bash script

```
#PBS -l select=1:ncpus=1:ngpus=1:mem=64GB:
   accelerator_model=rtx6000
```

Excercise 4.4 *Use of LLMs* (0 Points)

State for which exercise you have used LLMs (large language models) such as ChatGPT or GitHub Copilot. State which tools you have used and for which steps. This answer does not influence how many points you receive for your submission.