Miniproject BIO-322

Verest Mathieu
Romain Frossard

Team "The mid-table team"

EPFL

22 décembre 2023

Presentation of the project

The aim of the project is to develop a model for predicting the retention time (RT) of new drugs on a specific liquid chromatography machine based on their chemical characteristics. RT is crucial for drug identification and varies across different laboratories due to their specific chromatography setups and the chemical properties of the drugs being analyzed. Thus, the model aims to determine the RT of a specific molecule in a particular lab, replacing the need for experimental measurements.
To train our model, we utilized a large dataset containing the molecular structures of numerous drugs, including the SMILES format of the drug, circular fingerprint ECFP, and CDDD embeddings. Additionally, the dataset includes retention times measured in various laboratories using different chromatographic platforms.

Methods and Results

We commenced by gathering data from Kaggle, performing an initial examination and cleaning process to gain an initial understanding of the available data. Additionally, we established various metrics (contained in visual.ipynb) to guide our approach for future models and identify potential trends.
For detailed information on the data cleaning process, please refer to the GitHub readme and the accompanying code.
Subsequently, we created three datasets for evaluating our forthcoming model:

- [ ] CDDD: containing CDDD embeddings and one-hot encoded labels.
- [ ] ECFP: comprising ECFPs and one-hot encoded labels.
- [ ] Mix: a merged dataset incorporating both CDDD and ECFP data.

We employed two types of predictive models: linear regression (LR) and a forward neural network (FNN). Multiple configurations and parameter variations were tested to determine the most effective predictions. To attain the optimal model, we assessed the prediction efficacy using validation and cross-validation tests.

Linear regression

We implemented two types of regression: Ridge (L2 regularization) and Lasso (L1 regularization) regression. These methods were applied individually to the CDDD and ECFP datasets, as well as to the combined CDDD and ECFP dataset, to observe potential differences in results based on the utilized data.
It's important to note that in our implementation, we utilized nested cross-validation to robustly evaluate model performance, focusing on regularization and prediction while selecting optimal hyperparameters. We conducted various tests on our data, examining the impact of different modifications on model performance. Specifically, we experimented with standardizing (or not) the datasets (for mixed datasets and CDDDs), employing one-hot encoding for the lab data, and training the model using the logarithm of the RT. This decision was made after observing that the log-transformed output followed a normal distribution.

In the end, our best model was the Ridge linear regression on CDDD data, and obtained the following results:

- mean RMSE after CV = 1,180

Neural network

We initiated our work by constructing a feedforward neural network (FNN) using the PyTorch library. The initial model parameters were established, employing mean square error (MSE) as the loss function and Adam optimizer with a learning rate set to 0.001.

During the exploratory phase, we conducted experiments involving cross-validation and simple validation, adjusting various configurations. This included testing different hidden layer sizes (512, 1024, 2048, 5096), batch sizes, and tuning the number of epochs. Additionally, we experimented with activation functions like ReLU, Tanh, and sigmoid. After thorough testing, we identified the optimal settings as hidden layer sizes n1= 1024 and n2 =2048, a batch size of 64, 250-300 epochs (at which the test loss reached a plateau), and training on CDDD data.

However, we encountered a noticeable gap between test and train loss, indicating potential overfitting. To address this, we attempted regularization methods such as dropout and AdamW (acting as L2 regularization), which helped reduce the gap but increased the test loss. Seeking a suitable balance, we found that cyclically varying the learning rate during training, using the OneCycleLR technique, effectively regulated the model without compromising the test loss. Subsequently, we focused on combining ECFP and CDDD data as it yielded the most promising results.

With these new insights, we revisited both validation and cross-validation tests to fine-tune existing parameters and explore new ones, such as dropout rate, learning rate variations across epochs/batch, AdamW's weight decay, and batch normalization. Our primary objective was to achieve equilibrium between regularization and model performance.

Ultimately, our most successful model produced the following results:

- RMSE = 0.450
- Test loss = 0.2025
- Train loss = 0.108
- Kaggle score on 25% test set = 0.1900

(Detailed parameters available on GitHub and in the code)

Furthermore, we experimented with a network comprising three hidden layers, but unfortunately, this did not lead to performance improvements.

Conclusion

In our research, we adopted various methods to ascertain the most optimal prediction model. Initially, we conducted several types of linear regression, with Ridge regression on log-transformed and scaled CDDD data proving to be the most effective. Subsequently, we delved into exploring neural networks, implementing a dense neural network. The primary challenge lay in striking a delicate balance between performance, smoothness, and the prevention of overfitting. After experimenting with numerous strategies, we arrived at a model demonstrating considerable performance and apparent regularization, indicating the attainment of a suitable balance.

Overall, our best-performing model emerged as the neural network, significantly outperforming the linear regression approach.