Weizhen Liu

liuwzh@shanghaitech.edu.cn

Method

Data

We build models that given peptide sequence then output the predicted retention time and ion intensity associated with the peptide. The sequence is represented by the symbol of amino acids such as L, K, M, etc., typically 7-50 in length. Specially, we use 1 to represent the oxidation of methionine (M), and we use 2,3,4 to represent the phosphorylation of serine(S), threonine(T), tyrosine(Y), respectively. The retention time is a scaler, meaning the retention time of the peptide in the liquid chromatography. The ion intensity is the relative intensity (0-1) of ions fragmented in the mass spectrometry.

We have the dataset $\mathcal{D}$ composed of pair of $\{X, y\}$. $X := \{< x_1, x_2, x_3, \ldots, x_n >\}$, x_i is amino acid, and y is the retention time or ion intensity. We split the dataset $\mathcal{D}$ into train : validation : test by $8:1:1$. And we use validation set to tune the hyper-parameters and report the score on the test set.

Model method interpretation

It could be formulated as a regression problem from sequence to value: given sequence X, a learned function F could map the X to the y, either retention time or ion intensity. The mathematical expression is as follows:

$$\hat{y} = F(X; \Theta)$$

Θ is the parameters of the model.

We learn the parameters Θ by training dataset in $\mathcal{D}$, using root of mean squared error (RMSE) as well as mean squared error (MSE) as our loss function L_{RT} and L_{ion} for retention time and ion intensity prediction, respectively.

$$L_{ion} = \frac{1}{N}\sum_{i}^{N} \|\hat{y_i} - y_i\|^2$$

$$L_{RT} = \sqrt{\frac{1}{N}\sum_{i}^{N} \|\hat{y_i} - y_i\|^2}$$

Furthermore, we find the optimal parameters $\Theta^\star$ by minimize the loss function L.

$$\Theta^\star = \arg\min_{\Theta} L$$

We use the LSTM + Transformer model (illustrated in the Figure) to solve the retention time (RT) and ion intensity prediction task. LSTM is a kind of recurrent neural network(RNN) trying to use gate function to capture the long dependency in the input sequence. The bi-directional LSTM, expecting to learn a good token embedding for the network's downstream layers, is scheduled as the first module of our model. Transformer is the second module, an entirely different model by exploiting the self-attention compared to RNN. and it has achieved state-of-the-art performance in multiple natural language tasks, such as language translation, language entailment classification, language modeling, etc. We use the pre-layer form of transformer, which is proposed in (On Layer Normalization in the Transformer Architecture).

RT prediction

For this task, since the output of transformer is the same length as the input sequence, we need to take it down to one scaler for retention time. By adding the time distributed linear layer to assign varied weights for different amino acids dynamically, we use the weighted sum to obtain the final prediction. Pearson Correlation Coefficient (PCC) is used as metric, implemented in scipy.stats.pearsonr www.scipy.org and defined as:

$$PCC = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

m_x is mean of vector x, and m_y is mean of vector y. The $\Delta t_{95\%}$ metric is also used, which represents the minimal time window containing the deviations between observed and predicted RTs for 95% of the peptides.

$$\Delta t_{95\%} = 2 * |y - \hat{y}|_{95\%}$$

Ion intensity prediction

The model predicts four types of b ion and y ion for this task: ion charged one or two, combined with neu-

ral loss or one phosphate loss, so that we predict eight kinds of ions in total. We compute each peptide's PCC and select the median of those PCCs as the final evaluation metric, Primarily, we follow (Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning) using normalized spectral angle(SA) as another metric. Similarly, the median of those SAs is reported as the final evaluation metric.

Implementation details

The LSTM module comprises a stack of two units of two layers bi-directional LSTM, and transformer module is composed of n layer transformer encoder layer. For ion intensity prediction, n is 8. For retention time prediction, we implement five models, differing in the number of transformer encoder layer, with 4,5,6,7,8 layers of transformer encoder, then we independently train those 5 models. We average the results from those models as our final result. More model related configuration can be found on GitHub. We use Pytorch framework and train the model on TITAN Xp GPU.