Gene Editing with CRISPR-CAS9

NN and RNN

## Background and literature review

In the development of bioinformatics and biotechnology, neural networks play a vital role, due to the complexity of systems, which can mining and connect a large number of internal nodes from the information of input (Anderson, 1995). Meanwhile, be given weight through the linear and nonlinear relationship between every element, so as to getting Ideal output. (Chen Jie, 2015)

In the field of biological research, there is a direction that has received much attention in recent years, which is gene editing. The key point of gene editing is based on creating Double Strand Breaks (DSBs) at localization points within the genome. And the technical difficulties lie in precisely locating restriction enzyme’s cleavage sites. The most widely used tool in gene editing is CRISPR-Cas9, as an acquired immune system of prokaryotes, carries spacer RNA which helps to recognize and cleave disease-causing DNA. (Bae, 2018)

Traditionally, there are several traditional machine learning methods to deal with the regression problem. The most basic approach should be the linear regression, whose goal is to find a linear transformation from the input features to the output value. To deal with the overfitting problem of linear regression, ridge and lasso regression are proposed. Elastic net combines the benefit of ridge and lasso regression. However, linear regression cannot deal with the non-linear regression problems. There are also other methods like support vector regression based on the support vector machine, tree-based regression such as random forest / gradient tree boosting. However, both these traditional machine learning methods suffer from the feature selection problem. Feature engineering often becomes the most important work in the modelling of traditional machine learning models.

In order to improve the inference and prediction of guide RNA on-target efficacy, Neural network methods, especially Recurrent Neural Network (RNN), can structure predicted models for gene sequence (Chen Jie, 2015) . Unlike the Neural Network, RNN is mainly designed to process sequence data, mapping arbitrary length sequences into fixed length feature vectors. In the gene editing area, it mapping the encoded gene sequence data from input layer to hidden unit layer. (Chen Jie, 2015) In the hidden unit layer, it extracts features from current computed output and meanwhile it also has memory of previous information. The nodes between the hidden layers are connected, and the input of the hidden layer includes not only the output of the input layer, but also the output of the hidden layer at the previous time. Regarding the update of the weights, RNN uses the Backpropagation Through Time (BPTT) algorithm to reversely transfer the value of error at output using gradient descent (Nallapati, 2017).

According to Greff’s research (Greff, 2016), RNN may have problems with disappearance of gradient and gradient explosion. In order to solve these problems, various versions of RNN are proposed, such as GRU and LSTM. LSTM (long short-term memory) is a new kind of RNN. Therefore, this article also uses an improved method of RNN, Long-Short Term Memory algorithm (LSTM). LSTM constructs the structure of "gate", like input gate, forget gate and output gate, to increase or decrease the information of "cell state", which reserve the important content and remove the unimportant content.

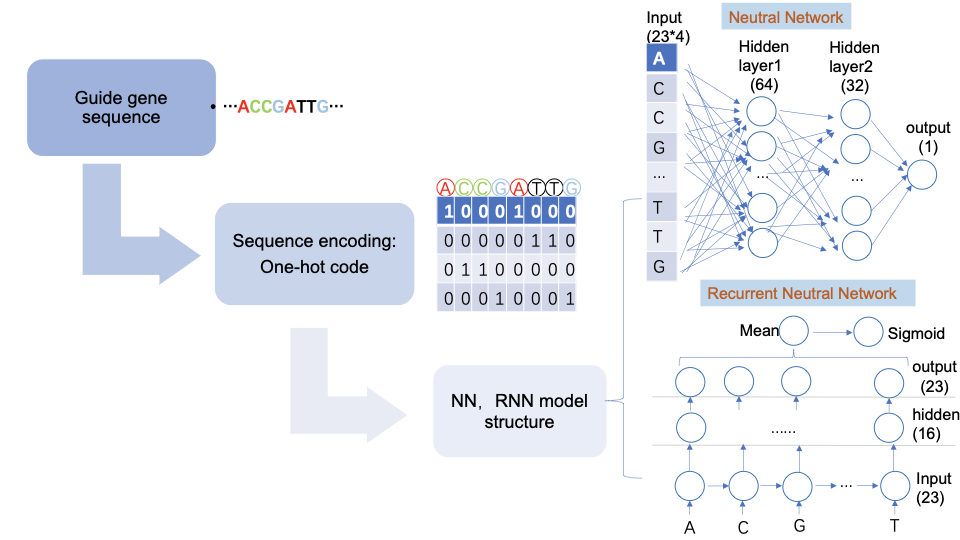
Accompanied by their powerful learning ability, the neural network and recurrent neural network often suffers from overfitting. There are some kinds of methods to overcome the problem of overfitting. Some of the most famous ones are regularization, dropout (Baldi, 2013), and early-stopping. Regularization adds some regularization factors in the loss function, such as L1 norm or L2 norm, to constrain the parameter size from increasing. Dropout randomly drops nodes of neural networks in the training process, it simulates a sparse activation from the input layer, which decreases the risk of overfitting from the beginning.

## Research questions

The purpose of this research is to predict the efficiency of the gene based on their sequences. The efficiency is a real number ranging from 0 to 100. Thus this is a traditional regression problem with a fixed value set and a sequence as the features.

## Methodology

The goal of our model is to predict the efficiency of the gene sequence, which is a value ranging from 0~99. This is a regression problem, we can use a raw neural network to solve it. (Specht, 1991)



As shown in the flowchart above, this model is mainly divided into the following steps.

Firstly, pre-processing and vectorizing the gene sequence. We divide the gRNA sequence dataset, which is processed by auto-encoder, into training dataset and test dataset. Then continuously divide the training dataset into training dataset and validation set according to the ratio of 8: 2.

Secondly, encoding the data. The one-hot encoding used here encodes each base in the sequence into a 1 \* 4 vector. The model specifies: A [1,0,0,0], T [0,1,0,0], C [0,0,1,0], G [0,0,0,1]. Since the length of gRNA is 23, 4 \* 23 matrix is​​obtained as the expression of each gRNA after encoding.

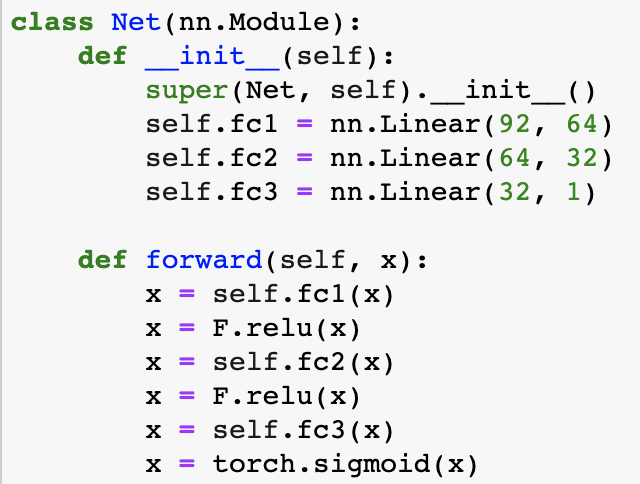
**Neural Network**

After preprocessing the data, we started to design the model. At first, we developed a traditional neural network model to predict gene sequences.

The NN model is mainly divided into an input layer, three fully connected layers and an output layer. The preprocessed 23 \* 4 gene sequence vectors are fed into the input layer. There are three fully connected layers in the middle, which is a linear combination of the feature dimensions of the previous layer and remapped to the new dimension. Beside that, we added a rectified linear unit function to each layer, so that we can not only linearly transform, but also learn nonlinear features. We experimented with several architecture of the neural networks, including one-layer neural network and two-layer neural network. For example, the configuration of two layer neural networks with hidden size=[64, 32], is explained as follows. The first layer, originally 92 dimensions, is mapped to 64 dimensions. The second layer, originally 64 dimensions, is mapped to 32 dimensions. The third layer maps from 32 dimensions to 1 dimension. In the last fully connected layer, a sigmoid activation function is set to normalize the output to (0, 1).

As for the loss function, the MSE (mean squared error) loss was chosen for our model. The MSE loss is the most commonly used loss function for regression. It measures the squared differences between true and predicted values.

As for the optimizer, we use SGD with momentum 0.9. After some attempts, I choose the learning rate of 0.1 in our experiments.



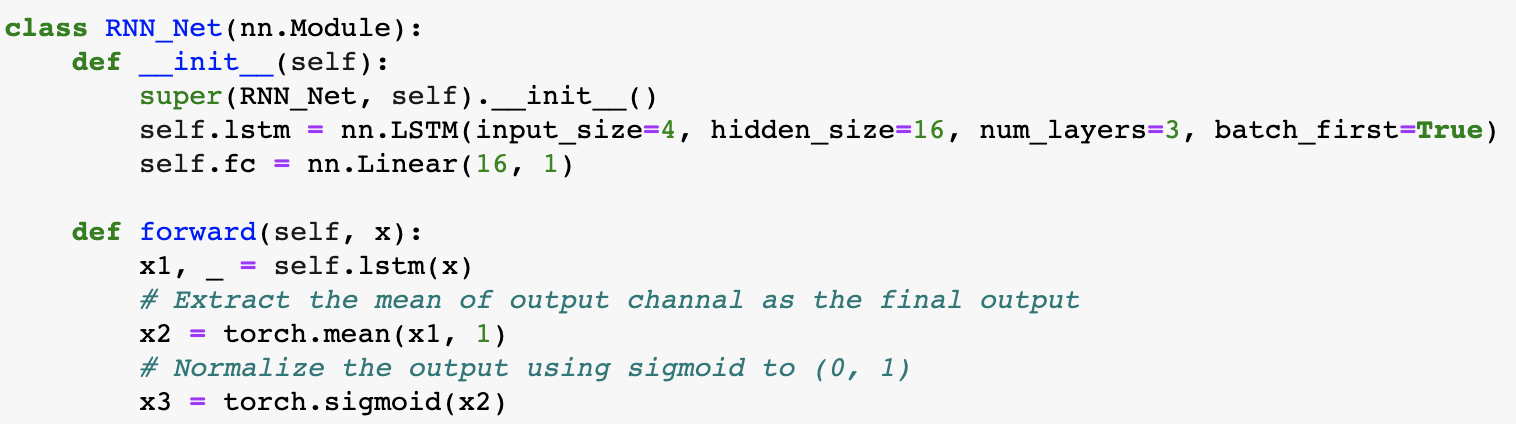
**Recurrent Neural Network**

Considering that there is some relationship between gene sequences. Analogous to the language model, there is a degree of relevance between each participle and its context in a sentence (Sundermeyer, 2012). Every word is greatly affected by the context, so for the sequence problem in this experiment, we developed an RNN model to make predictions.

The structure of RNN has the following parts. Similarly, the input of the RNN is also 4 \* 23 dimensions gene sequence vectors, but the difference is that the model will be fed 1 base vector (1 \* 4) each time, and each input sequentially updates the state of the middle layer.

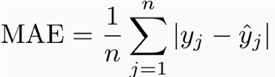
The parameters mainly include the size and number of hidden layers. Firstly, we set the dimension of the middle layer to be 16. the nodes and weights of the hidden layer are iteratively updated through successive input. We then extract the mean value of the output channel as the final output. Also as a regression problem to a fixed set [0, 1], we utilize the sigmoid function as a normalized function, to turn the output into [0, 1].

We also use MSE loss for the RNN model.

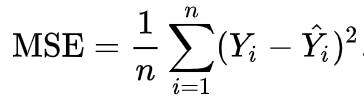
****

**Evaluation Methods**

**MAE: mean absolute error**

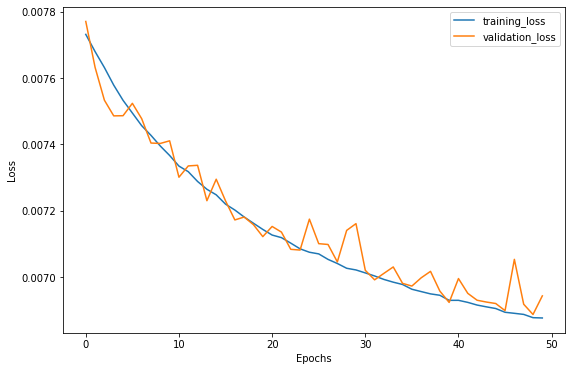
****

**MSE: mean squared error**

****

As evaluation methods, we use MAE and MSE as a criterion. Both are very common evaluation methods in regression problems. MAE measures the mean absolute error according to the predicted value and the true label, while MSE measures the mean squared error.

**Results**

****

The raw neural network

|  |  |  |
| --- | --- | --- |
|  | MSE | MAE |
| hidden size=16 | 23.7 | 3.94 |
| hidden size=32 | 22.5 | 3.72 |
| hidden size=64 | 20.5 | 3.48 |
| hidden size=[32, 16] | 21.1 | 3.50 |
| hidden size=[64, 32] | 19.9 | **3.41** |

The recurrent neural network

|  |  |  |
| --- | --- | --- |
|  | MSE | MAE |
| Layer=1, hidden size=16 | 70.9 | 6.47 |
| Layer=1, hidden size=64 | 68.5 | 6.29 |
| Layer=2, hidden size=16 | 67.8 | 6.25 |
| Layer=2, hidden size=64 | 65.5 | 6.17 |

Discussion and conclusions

In this research, we have investigated the performance of the raw neural networks and the recurrent neural networks to predict the gene efficiency based on their sequences. The results showed that both the raw neural network and the recurrent neural networks can accomplish the task of predicting gene efficiency.

Generally speaking, the raw neural network performs better than the recurrent neural network. This may indicate that although the RNN is commonly used in sequence modelling, the gene sequence however does not imply any sequence-dependent information in it, thus a raw neural network should satisfy our requirement. We also hypothesize that using more data could train a good RNN model.

The best configuration of the neural network achieved **MAE = 3.41**, as the original range of efficiency is 0~99, a MAE of 3.41 is quite a remarkable result to show, which shows the network performs pretty well on the test set. And according to the training / validation loss curve as shown above, the model does not suffer from overfitting or underfitting in the dataset. I also noticed that the training loss is quite close to the test error, showing the model generalize quite well on the test set.

For the parameter choosing of neural network, the results show that a larger hidden size and a deeper architecture should be beneficial to the performance. To be more specific, a hidden size of 32 is better than a hidden size of 16. And a two-layer neural network with the same hidden size is better than a single-layer one. This is quite reasonable because a deeper network means more parameters the model can learn and more complicated the model is. As the training set contains more than 200000 data, a two-layer network is proved to perform the best. I also tried some regularization methods to improve the results on the test set, such as dropout and L2 norm, but saw no performance gain.

In conclusion, for the gene prediction problem, the raw neural networks performs better than the recurrent neural network. The best MAE on the test set hit 3.41 with the range from 0~99, which shows the great generalization ability of the raw neural networks.

Reference

Anderson, J.A., 1995. *An introduction to neural networks*. MIT press.

Bae, S. and Kim, J.S., 2018. Machine learning finds Cas9-edited genotypes. *Nature biomedical engineering*, *2*(12), pp.892-893.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R. and Schmidhuber, J., 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), pp.2222-2232.

Peng, Z., Chen, J., Cao, L. *et al.* Identification of TSS in the human genome based on a RBF neural network. *Int J Automat Comput* **3,** 35–40 (2006). https://doi.org/10.1007/s11633-006-0035-7

Nallapati, R., Zhai, F. and Zhou, B., 2017, February. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Sundermeyer, M., Schlüter, R. and Ney, H., 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

Specht, D.F., 1991. A general regression neural network. *IEEE transactions on neural networks*, *2*(6), pp.568-576.

Baldi, P. and Sadowski, P.J., 2013. Understanding dropout. In *Advances in neural information processing systems* (pp. 2814-2822).