PROTONET – SUPPLEMENTARY MATERIAL

LEANDER DONY[1]

June 2017

CONTENTS

LIST OF FIGURES

[1] *Department of Life Sciences, Imperial College London, United Kingdom*

1 PERFORMANCE METRICS

We have used several different performance metrics to evaluate our models. Metrics were calculated as outlined below using the following definitions:

TP: True positive (a positive prediction which is actually true)
FP: False positive (a positive prediction which is actually false)
TN: True Negative (a negative prediction which is actually false)
FN: False Negative (a negative prediction which is actually true)

1.1 Accuracy

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

1.2 Precision (Positive Predictive Value)

$$PPV = \frac{TP}{TP + FP} \tag{2}$$

1.3 Recall (True Positive Rate)

$$TPR = \frac{TP}{TP + FN} \tag{3}$$

1.4 False Positive Rate

$$FPR = \frac{FP}{FP + TN} \tag{4}$$

1.5 Matthews Correlation Coefficient

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

1.6 F1 Score

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN} \tag{6}$$

2 DETAILS: INPUT DATA

2.1 Protein Interaction Data

The protein interaction data obtained from the Biogrid contained 16,109 human-only proteins with 219,216 interactions between after removing protein self-interactions.

From the Biogrid PPI data, we constructed a network using the NetworkX Python package. Using multidmesnional scaling (MDS), we embedded the network data in a four-dimensional space using the approach proposed by

Kuchaiev et al. (2009). The provided Matlab Code was translated to Python to enable compatibility with the constructed NetworkX graph.

2.2 Protein Domain Data

The protein domain information was originally obtained as a multi-label matrix of the shape 16109 x 11664 (number of proteins x number of domains), where each '1' in the column would represent the presence of a specific domain in the respective protein. In total, the domain data included 56358 positive labels with 5.7 % (922) of the 16109 proteins having no domain label.

2.3 Gene Ontology Data

We were able to obtain molecular function GO data for 28 % (4,473) of the 16,109 proteins and biological process data for 27 % (4,389) of them. 71 % (11,438) of the proteins had neither molecular function nor biological process GO terminology available.

3 TRAINING, TEST AND VALIDATION SETS

We created a test and validation set by randomly removing 20 % of all positive samples (edges) from the PPI network. We then removed these interactions from the input data and split the removed interactions in two equally sized groups (test and validation). Additionally, we moved a further 10 % of the negative samples (non-interacting protein pairs) from the input data to each the training and validation data. As a result we had created a training, validation and test set with equal distributions of positive and negative samples and a split of $0.8 : 0.1 : 0.1$.

For the final performance test of the data, we reintroduced the edges removed for the validation set to the network and reran the embedding so that we could train the network on the combined training and validation data before testing.

An alternative to the single-fold validation approach would be a more thorough cross-validation process. Here the training data is divided up into for example ten folds, each of which is used to validate the model in turn. Given the computational cost of this process, this is usually not a viable option for the optimisation of neural networks.

4 MEMORY CONSIDERATIONS

Following the reduction of the input dimensions, we obtained a coordinate input of 4 dimensions, a domain input of 512 dimensions and a GO terminology input of 512 dimensions. Considering, that there are $n * (n-1)/2$ possible interactions, between 16,109 proteins, we obtained 129,741,886 samples of input data. Assuming 4 bytes of occupied memory per single number (float32), and between 2056 dimensions per sample, the memory required to hold all input data at once would have been 1 TB (or 22 TB without dimensionality reduction using the autoencoder). This amount of data could not all be held in memory at once given the hardware available for this project.

5 NEURAL NETWORK ARCHITECTURE

In the "General Network Structure" section on the website, we discussed the broad outline of how sets of hidden layers are aligned in our neural network model. Here, we describe the detailed structure of the hidden layer set employed in our model.

Each set consists of one or more or more layers (with 1024, 128 or 64 nodes). Each layer in turn is comprised of multiple sublayers (definions from the Keras documentation[1]):

1. Dense layer with L2 regularisation and *he_normal* weight initialisation
2. Batch normalisation layer
3. ReLU activation layer
4. Dropout layer

As a result, a three-input model with three layers per set would have a total of 12 Dense layers, 12 BatchNorm layers 12 ReLU layers and 12 Dropout layers (if dropout rate set to is greater than 0.0) arranged in the previously discussed order.

6 NEURAL NETWORKS: DEFINITIONS

6.1 Activation Functions

The three nonlinearities ("activation functions") used with the neural networks in this project are shown in figure 1. They are mathematically defined as follows:

ReLU

$$f(x) = \max(0, x) \tag{7}$$

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

Tanh

$$f(x) = \tanh x = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{9}$$

1 https://keras.io/layers/about-keras-layers/

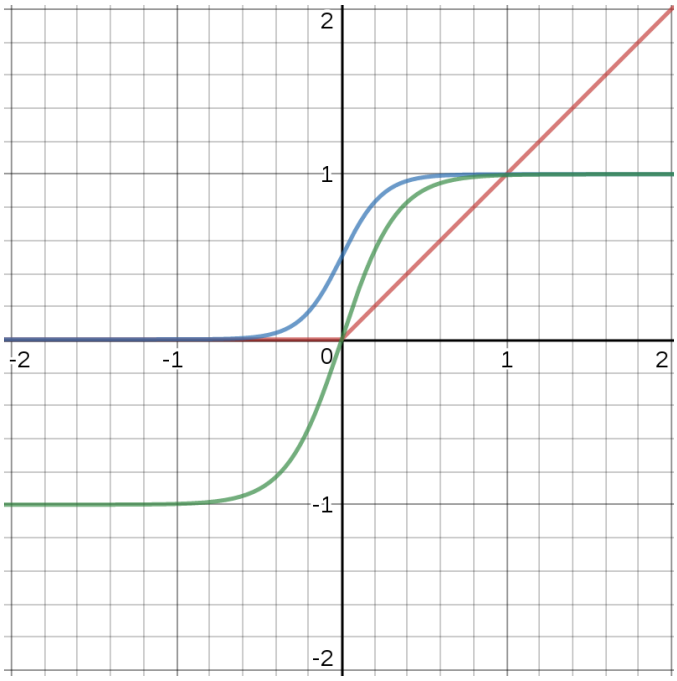

Figure 1: Three common activation functions used with Artificial Neural Networks: ReLU (red), sigmoid (blue) and tanh (green).

6.2 Loss Functions

We used the binary crossentropy loss together with an L2 regularisation for all models.

The total loss is therefore defined as follows:

$$L = \underbrace{\frac{1}{N}\sum_{i} L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}} \tag{10}$$

Where N is the number of samples, λ is the regularisation strength and W are the weights of the model.

The data and regularisation loss are defined as follows:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{11}$$

$$R(W) = \sum_k \sum_l W_{k,l}^2 \tag{12}$$

6.3 Optimisation

Minimising the loss function can be understood as a convex optimisation problem. We have used two different optimisers for this purpose: the Adam optimiser (Kingma & Ba 2014) and stochastic gradient descent with Nesterov momentum (Nesterov 1983). Please consult the respective original publication for details on the underlying maths and logic.

REFERENCES

Kingma, D. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kuchaiev, O., Rašajski, M., Higham, D. J. & Pržulj, N. (2009), 'Geometric de-noising of protein-protein interaction networks', *PLOS Computational Biology* (8), e1000454.

Nesterov, Y. (1983), A method of solving a convex programming problem with convergence rate o (1/k2), *in* 'Soviet Mathematics Doklady', Vol. 27, pp. 372–376.