January 20, 2024

1 ANN

1.1 Fully connected neural network

Fully connected neural network, also known as Multilayer Perceptron (MLP), is one of the most common neural network models.

The leftmost layer is the input layer, which is responsible for receiving data and passing it to the next layer. The middle layers are called the hidden layers, and the rightmost layer is called the output layer. The hidden layers and the output layer are responsible for processing and outputting data. The number of layers is the depth of the neural network, and the number of neurons in the hidden layer is the width of the neural network.

The connections between neurons in each layer have connection weights. And there are biases on each neuron in the hidden layer and the output layer. Learning process of the neural network is to adjust these weights and biases through the training data, so that the model can fit the data well. Therefore, it can be considered that what a neural network has learned is implicit in these weights and biases. Let the data received by the input layer be X, then the data passed to the hidden layer after the transformation is $\sigma(WX + b)$, where W is the weight matrix and b is the bias vector. σ is the activation function, which introduces nonlinearity in the model. The commonly used activation functions are sigmoid, tanh, ReLU, etc. Their expressions are as follows:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

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

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{3}$$

When modeling with neural networks, it is very important to determine the structural parameters (width, depth) and training parameters (learning rate, etc.) of the neural network. Although neural network structural search technology and Bayesian optimization methods are available, these methods often require more sophisticated technical support and higher computational resources,

and there is uncertainty in the results. Therefore, here we still determine the values of each parameter based on experience and testing on a small data set. The neural network we use in this paper contains four hidden layers, each with $128, 64, 64, 64$ neurons. The number of neurons in the input layer is equal to the number of input features, while the output layer has only one neuron. The activation function between hidden layers is LeakyReLU .

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \lambda x, & \text{if } x < 0 \end{cases} \tag{4}$$

Where λ is an adjustable hyperparameter, the proposer of this function suggests taking it as 0.01 or even smaller, which is set to 0.2 in this paper by testing on the dataset. The activation of the hidden layer to the output layer is linear activation, i.e., $y = x$.

Loss function is another important factor to consider in the neural network modeling. When using neural networks for regression tasks, the commonly used loss functions are mean absolute error (MAE) or L1 loss, mean square error (MSE) or L2 loss as follows:

$$L_1(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{5}$$

$$L_2(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{6}$$

These equations define two different loss functions used in statistics and machine learning to quantify the difference between the predicted values $\hat{y}_i$ and the actual values y_i:

1. L_1 Loss Function (also known as Mean Absolute Error or MAE): It calculates the average of the absolute differences between the predicted values and actual values. This loss function is less sensitive to outliers compared to the L_2 loss function.

2. L_2 Loss Function (also known as Mean Squared Error or MSE): It calculates the average of the squares of the differences between the predicted values and actual values. This loss function is more sensitive to outliers because it squares the differences, which disproportionately affects larger errors.

The predicted values of the model have to satisfy certain physical constraints

while being accurate. Therefore, we design a loss function of the following form:

$$\begin{aligned}\text{Loss}(\mathbf{y}, \hat{\mathbf{y}}) =& a_1 \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right) + \\ & a_2 \left(\frac{1}{n} \sum_{i=1}^{n} \max(0 - \hat{y}_i, 0) \right) + \\ & a_3 \left(\frac{1}{n} \sum_{i=1}^{n} \max(\hat{y}_i - 1, 0) \right) + \\ & a_4 \left(\frac{1}{n} \sum_{i=1}^{n} \max(y_i - \hat{y}_i, \beta y_i) \right) + \\ & a_5 \left(\frac{1}{n} \sum_{i=1}^{n} (\tau_i - \hat{\tau}_i)^2 \right)\end{aligned} \tag{7}$$

where $a_i (i = 1, 2, 3, 4)$ is the proportion of the error in each component, which can be adjusted according to the actual situation.

- $\mathbf{y}$ and $\hat{\mathbf{y}}$ represent the actual and predicted values, respectively.
- a_1 to a_5 are coefficients that weight the different components of the loss function.
- The first term is the mean squared error.
- The second term penalizes predictions $\hat{y}_i$ that are less than zero.
- The third term penalizes predictions $\hat{y}_i$ that are greater than one.
- The fourth term is a customized loss that takes the maximum of the difference $y_i - \hat{y}_i$ and βy_i, which might be a form of regularization or constraint.
- The fifth term appears to be another mean squared error term, potentially for a different set of variables τ_i and $\hat{\tau}_i$.
- The index n is the number of samples or predictions.

In the above equation, the first term is the MSE loss of data. The second term represents the loss arising from the fraction of model predictions less than 0. Here, a priori knowledge is that the eddy viscosity will not be a negative value. Therefore, it is necessary to put some constraints on the negative values in the model predictions. When the prediction is negative, the second term generates losses to guide the model to learn in a more physically consistent direction. The third term has a similar meaning to the second term and serves to constrain the model to produce predictions greater than one. This is mainly because we normalize the data to the range between 0 and 1. The fourth term

is a relative error between the predicted and true values, which is constructed based on the a priori that "the eddy viscosity coefficient inside the boundary layer is small but more important". β is an adjustable constant that indicates the magnitude of the tolerable relative error. The last error is the error about Reynolds stress, where the stress term is calculated according to the following formula.

$$\tau = 2\mu \|S_{ij}\| \tag{8}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \tag{9}$$

where $\|S_{ij}\|$ is the norm of the strain rate tensor S_{ij}. This loss is also added to highlight the importance of the eddy viscosity inside the boundary layer and to constrain the network learning process from the perspective of Reynolds stress. This loss also ensures that the coupled iterative solution proceeds smoothly to a certain extent since it is not the eddy viscosity coefficient but the stress term that is involved in the coupled calculation.

- The $\|S_{ij}\|$ denotes the norm of the strain rate tensor S_{ij}.
- μ represents the dynamic viscosity.
- τ represents the stress tensor.
- U_i and U_j are the velocity components.
- x_i and x_j are the spatial coordinates.

By adjusting the coefficients in front of each loss term, the model can be targeted to learn in a certain direction. In the reference the authors set it to 0.6, 0.1, 0.1, 0.1, 0.1, respectively. In a specific training process, the above five losses are at the same magnitude, and they decrease consistently as the training progresses.

The neural network model used in the reference is based on Pytorch(Paszke A, Gross S, Massa F, et al. Pytorch: an imperative style, high-performance deep learning library[J]. arXiv preprint; 2019. arXiv:1912.01703.).

The optimization algorithm for training is the Adam algorithm(Kingma DP, Adam Ba J. A method for stochastic optimization[J]. arXiv preprint;2014. arXiv:1412.6980.) with batch size of 128 and an initial learning size of 0.003.

As the training continues and the error decreases, we use a dynamic adjustment to reduce the learning rate. The specific method is to change the learning rate to the original 0.5 when the model's loss on the validation set no longer decreases. Otherwise, training epoch is 300. When the performance of the model on the training set and the validation set tends to be stable, the error is on the order of 10^{-5}.