MSBD5002x Project

Classification of COVID-19 Infection Based on Neural Network

Tuo Xiong
2274229298@qq.com

1. Objective

This is a project involving in the application of neural network on real-world classification problems, which includes two main objectives.

The first objective is to apply data mining techniques on the given two datasets, which contain information and features of the confirmed or potential COVID-19 infected, and try to distinguish how likely an individual is infected given certain features. The expected result will be that given new features of uncertain infection, this program (based on python3) is supposed to output a credible binary prediction on if an individual is infected or not. Specifically, an 1 will be regarded as infected and a 0 will not.

The other objective is to compare and evaluate the model performance of neural network with different structure and combination of parameters. Specifically, the variables include the amount of hidden layers, the amount of neurons in each layer, the choice of activation function, loss function, optimizer, the percentage of validation set, the number of epochs and batch size.

2. Datasets

The two original datasets in this program are “firstData.txt” and “secondData.txt”. The first one includes 9 binary features (Fever, Tiredness, Dry-Cough, Difficulty-in-Breathing, Sore-Throat, Pains, Nasal-Congestion, Runny-Nose, Diarrhea), 3 categorical features (Age, Gender, Country) and 1 target binary feature (Infected). Therefore the categorical features are transformed into numeric values by label encoding. For the feature Age, the 5 unique categories of age ranges are transformed to integer 0-4 respectively, and similarly the feature Gender 0-2 and Country 0-9. Then we split out the last column of target feature as the labels “y” for model training. The final input dataset used for training which is denoted by upper “X” includes 12 numeric feature and the preprocessing is finished here.

Besides, the second one is a test set without target feature given. The pipeline of transformation is stored in this program so the manipulation on the dataset of new input features is exactly the same as that on the first dataset. Here we let “newX” denote the transformed new input dataset. In the final step, the “newX” which also includes 12 numeric features will be fed to the program and output the prediction for test set.

3. Models

In this program, 5 neural network models will be saved and used for further prediction according to the customized variables. Indeed, the output of the program is not limited to 5 models but more as long as users define the variables or parameters of models well. All models use the same transformed “X”, “y” and “newX” mentioned in the last section for training and test prediction. For my individual favor, the parameter combination of these 5 models is as the following:

i. 12 neurons in the first hidden layer with Rectified Linear Unit activation function (ReLU); 8 neurons in the second hidden layer with ReLU; 1 output neuron with Logistics Sigmoid activation function (Sigmoid); binary cross-entropy as loss function; Adaptive Moment Estimation (Adam) as optimizer; 20% training set into validation set; 50 epochs and a batch size of 10.

ii. 4 neurons in the first hidden layer with ReLU; 2 neurons in the second hidden layer with ReLU; 1 output neuron with Sigmoid; binary cross-entropy as loss function; Adam as optimizer; 20% training set into validation set; 30 epochs and a batch size of 30.

iii. 4 neurons in the first hidden layer with ReLU; 2 neurons in the second hidden layer with ReLU; 1 output neuron with Sigmoid; binary cross-entropy as loss function; Stochastic Gradient Descent (SGD) as optimizer; 20% training set into validation set; 30 epochs and a batch size of 30.

iv. 4 neurons in the first hidden layer with Sigmoid; 2 neurons in the second hidden layer with Sigmoid; 1 output neuron with Sigmoid; binary cross-entropy as loss function; Adam as optimizer; 20% training set into validation set; 30 epochs and a batch size of 30.

v. 8 neurons in the first hidden layer with ReLU; 1 output neuron with Sigmoid; binary cross-entropy as loss function; Adam as optimizer; 20% training set into validation set; 50 epochs and a batch size of 10.

4. Results

Due to the definition of random seed on current time, the results of models may be slightly different among different runs. The accuracy of model 1 in validation set is always 100%. The accuracy of model 2 in validation set is very close to 100% due to less epochs and a larger batch size. The accuracy of model 3 in validation set is unstable from over 60% to 90%, because the default SGD method may randomly pick the initial weights and eventually reach a local minimum instead of the global minimum. The accuracy of model 4 in validation set is between 90% and 100%. The accuracy of model 5 in validation set is also exactly or very close to 100%.

The model 1 has 2 hidden layers, relatively more neurons in each layer, more epochs, a smaller batch size and utilizes only ReLU activation function for hidden layers. These features make it perform the best among these 5 models. Models 2-5 all have disadvantage compared with model 1. In model 2, less neurons, epochs and a larger batch size mean the result may not have sufficient renewal steps to converge. In model 3, the convergence speed of SGD method seems to be slower than Adam method. In model 4, the performance of Sigmoid activation function on hidden layers seems worse than ReLU. In model 5, there is only 1 hidden layer left. It still gets a satisfying result although the convergence speed is slower than model 1. The accuracy of model 1 converges approximately at the 14th epoch but the accuracy of model 5 converges at the 27th epochs.

5. Conclusion

In this classification problem, the original datasets are clean. The binary and categorical features are also well prepared and suitable for model training. Under such condition, the performance of multi-layer neural network with Adam optimizer is sufficient and will not be affected much by the quality of datasets. In fact, this program does provide extremely accurate prediction in the validation set. However, if the quality of datasets is low, the result may depend much more on data preprocessing. Overall, the performance of neural network model is good enough in this case, as long as suitable parameters or variables are chosen for the structure of the model.