**The Instruction of Task1 Assignment to Chuhan Xu**

By Xia Jiang

2020.12.18

Run prediction methods for all datasets provided by Dr. Jiang (shared with you via BOX), prioritized in the following orders: distance recurrence, survival, survival grouped, and PCP. Prediction methods including the following: Deep learning, *Naïve Bayes* (*NB*), *logistic regression* (*LR*), *decision trees*, *support vector machines* (*SVM*) with the r*adial basis function* (*RBF*) kernel, the *least absolute shrinkage and selection operator* (*LASSO*), *k-nearest neighbors* (*KNN*) , and the *extreme learning machine* (*ELM*). All methods should be implemented and tested in two environments: python and R as a cross check in the following orders: Python first and then R. When implementing deep learning in python, you should start out with pyTorch and then Keras.

A dataset will be split into a train-validation set, containing 80 percent of the cases, and an independent test set, containing the remaining 20 percent. The dataset division is mostly done randomly except that the independent test set will have approximately 20% of the positive cases and 20% of the negative cases to ensure the test set represents all patient states. We then perform a 5-fold cross validation (CV) to predict the binary outcome. Each of the five folds has a train set containing 80 percent of the train-validation set and a validation set containing the remaining 20 percent. For each fold, AUCs will be computed for every set of hyperparameters via a gird search with all possible combinations of parameter values. The AUCs corresponding to each set of hyperparameters will be averaged over all five folds with each set of hyperparameters and the best-performed set of hyperparameter values were chosen based on the highest mean AUC, with which we will apply the method one more time to learn the selected prediction model using the entire train-validation set. We will then apply that model to predict each of the cases in the 20 percent of the independent test set that was set aside. In each test set, we will predict all the positive outcome cases and all the negative cases; doing so will yield counts of the correctly predicted and incorrectly predicted, and with which to derive the final performance measure (AUC).

For each method with each dataset, learn the best set of parameter values guided by 5-fold CV by trying all possible sets of parameter values. You can start out with suggested values at the end of this file. You should continue to try other values and different optimization methods after obtaining results with the initial suggested values to reach the optimal results. You will generate the AUC score and create AROC curve for the top set. In addition, for each experiment generate the following output: running time in a summary time table and a readme file which should include date and time when experiment starts and ends, dataset, method, parameters, and compute hardware information. These results are expected:

1. The raw result for each experiment, well organized with all files properly described/labeled for easy retrieval.
2. A summary table for each of the parameters involved in your tuning process and all values that you tried, including all methods. An read me files for each experiements.
3. A summary table for the best AUC score and the best set of parameter values, from which this score was derived, for each of the methods using each dataset.
4. A summary table for all the running time recorded.
5. Proper labeled AROC curves.

2 through 5 should be well organized and uploaded in the shared BOX folder called ExperimentResults.

All source code should be well organized and uploaded in the shared BOX folder called code.

Any subfolders and file should be named in a structured and meaningful way so that they can be retrieved easily.

Please updated the BOX folders each day with the work you are done in that day!

|  |  |
| --- | --- |
| LAY: # hidden layers | HID: # hidden nodes in each layer |
| 1 | 10,000; 5000; 3500 2500; 2000 1500 1250; 1000, 750, 500; 250; 100; 50; 25; 10 |
| 2 | 5000; 4000, 3500, 3000, 2500, 2000, 1500, 1250; 500; 250; 100; 50; 25; 10 |
| 3 | 2500; 1250; 500; 250; 100; 50; 25; 10 |
| 4 | 1500; 1000, 500; 250; 100; 50; 25; 10 |

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Description** | **Values** |
| Depth | Number of hidden layers | See Table 1 |
| Number of nodes in each hidden layer |  | See Table 1 |
| Number of nodes in input layer |  | See Table 1 |
| Epochs | Number of times model is exposed to entire training data | {20, 1000} |
| Weight initialization | Heuristic used to initialize weights prior to training [12] | {he\_normal, glorot\_normal} |
| Training optimizer | Learning algorithm or updater that minimizes a loss function and updates model parameters accordingly [9, 12, 15] | {sgd, adagrad} |
| Learning rate | Used by optimizer; affects speed of training [8, 9] | {0.1, 0.01} |
| Momentum | Attenuates oscillation of weights during training [21]. *Not used by adagrad optimizer* | {0, 0.5, 0.9} |
| Decay | Regularization technique; weights are multiplied by decay factor after every update [13] | {0, 0.0001, 0.0005} |
| Dropout rate | Regularization technique; reduces overfitting and coadaptation by removing nodes [17] | {0, 0.2, 0.4, 0.5} |
| L1 | Sparsity regularization [18] | {0, 0.01} |
| L2 | Weight decay regularization [19] | {0, 0.01} |

***Support vector machines***

For SVM Linear Kernel, you can first try these values of parameter C: 2-5, 2-1, 23, 27, 211

For SVM RBF Kernel, you can first try these values of parameter C: 2-5, 2-1, 23, 27, 211, and these of parameter ϒ: -11, -7, -3, 1, 5, and take all 25 combinations of the parameters.

For other methods, please start out with parameter values suggested either by user guide (manual) or literature.