Supporting Information_2

Classification of HIV-1 Protease Inhibitors by Machine Learning Methods

Yang Li[1,2], Yujia Tian[1], Zijian Qin[1], Aixia Yan[1]*

1. State Key Laboratory of Chemical Resource Engineering, Department of Pharmaceutical Engineering, P.O. Box 53, Beijing University of Chemical Technology, 15 BeiSanHuan East Road, Beijing 100029, P. R. China.

2. Institute of Science and Technology, Shandong University of Traditional Chinese Medicine ,Ji'nan, Shandong , 250355,　China

*Corresponding author: yanax@mail.buct.edu.cn，phone: +86-10-64421335

The process of optimizing parameters of different machine learning models

1.The process of optimizing parameters of k-Nearest Neighbors (K-NN) models

The main idea for k-Nearest Neighbors (K-NN) is that if a sample is in the most contiguous sample in the feature space, most of them belong to a certain category, and also this sample belongs to this category. The algorithm averages the response value over the closest neighbors to estimate the response value. There is one important parameters, the number of the closest neighbors (n). The optimization of the parameter can be taken out by a cross-validation with the python script which used the packets “sklearn”, imported “KNeighborsClassifier” of python.

Details on optimum parameters are as follows:

First, the parameters s and d were optimized by the python script with ergodic search method based on the training set of each model.

Second, based on the “KD_TREE” algorithm with a ergodic step of every parameter is 1, parameter n was from 3 to 10, leaf_size was set up as 20, 30, 40, 50 and weights was set up as “uniform”, so we got several groups of parameters.

Next, every group of parameters was used for building models by a 5-fold validation method on a corresponding training set to calculate mean squared error (MSE). Only one group of parameters with the minimal MSE value was used as optimum parameters of K-NN model.

2.**The process of optimizing parameters of Decision Tree (DT) models**

For a Decision Tree (DT), each non-leaf node contributes to a test to be conducted on a single attribute value, each branch represents the attribute in a certain range of output, and each leaf node storing a category. We used the Classification And Regression Tree (CART) algorithm for modeling. CART provides binary trees that output the largest information gain at each node. There are two important parameters, the minimum number of samples (s) and the range of the maximum depth (d) in CART algorithm. The optimization of these two parameters can be taken out by a cross-validation with the python script , which used the packets "sklearn", imported "DecisionTreeClassifier" of python.

Details on optimum parameters are as follows:

First, the parameters s and d were optimized by the python script with ergodic search method based on the training set of each model.

Second, with a ergodic step of every parameter is 1, parameter s was from 1 to 5, parameter d was from 0 to 12, and max_features was set up as "auto", so we got $5\times 13=65$ groups of parameters.

Next, every group of parameters was used for building models by a 5-fold validation method on a corresponding training set to calculate mean squared error (MSE). Only one group of parameters with the minimal MSE value was used as optimum parameters of this DTmodel.

3.The process of optimizing parameters of Random Forest (RF) models

A Random Forest (RF) is an ensemble of unpruned classification trees created by using bootstrap samples of the training data and random feature selection in the tree induction. The essence of a RF combines multiple decision trees depending on an independent drawing of samples to improve the result of the Decision Tree. There are two important parameters, the number of trees (n) and the range of the maximum depth (d) in RF. The optimization of these two parameters can be taken out by a cross-validation with the python script , which used the packets “sklearn”, imported “RandomForestClassifier” of python.

Details on optimum parameters are as follows:

First, the parameters n and d were optimized by the python script with ergodic search method based on the training set of each model.

Second, with a ergodic step of every parameter is 1, parameter n was from 10 to 150, parameter d was from 0 to 20, and max_features was set up as “auto”, so we got 141×41=2961 groups of parameters.

Next, every group of parameters was used for building models by a 5-fold validation method on a corresponding training set to calculate mean squared error (MSE). Only one group of parameters with the minimal MSE value was used as optimum parameters of this RFmodel.

4.The process of optimizing parameters of support vector machine (SVM) models

The brief principle of SVM is shown via the equation (1) and (2).

$$\forall \varepsilon > 0,\ \exists\ f(x) = \omega^T \cdot x + \omega_0 \quad (1)$$

$$\text{subject to: } \left| y_i - f(x_i) \right| - \varepsilon \leq 0 \quad (2)$$

where ε represents the insensitive loss parameter of a support vector machine, ω and ω_0 represent the weights of a support vector machine, i represents the index of input data, x_i and y_i represent the attributes and a label of input data, respectively.

The epsilon-SVR (ε - support vector regression) which was one of the SVM-regression methods was adopted in this work and the aim of epsilon-SVR is shown via the equation (3) and (4).

$$\min_{\omega,b,\xi,\xi^*} \left[\frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}(\xi + \xi^*)\right] \qquad (3)$$

$$\text{subject to} \begin{cases} y_i - f(x_i) \leq \varepsilon + \xi \\ f(x_i) - y_i \leq \varepsilon + \xi^* \\ \xi, \xi^* \geq 0 \end{cases} \qquad (4)$$

where C represents the loss parameter, ξ and ξ^* represent the relaxation factors of a support vector machine, respectively.

The kernel functions are very important in SVM-regression. There four kernel functions, which include linear function, polynomial function, radial basis function (RBF) and sigmoid function, are available in LibSVM toolbox.

According to equation (1) - (4), there are four important parameters, C, g (kernel function parameters), ε and kernel function type in epsilon-SVR analysis. The optimization of these four parameters can be taken out by the python script "gridregression.py" which is available at the LibSVM website. The "gridregression.py" uses the ergodic search with a cross validation method to optimize parameters C, g and ε in a given kernel function type.

Details on optimum parameters are as follows:

First, the parameters C, g, ε were optimized by the python script “gridregression.py” with ergodic search method based on the training set of each model.

Second, with a ergodic step of every parameter is 2^1, parameter C was from 2^{-10} to 2^{10}, and parameter g was from 2^{-10} to 2^{10}, and parameter ε was from 2^{-10} to 2^{-10}, and the type of kernel functions was radial basis function (RBF), so we got 20^3 groups of parameters.

Next, every group of parameters was used for building models by a 5-fold validation method on a corresponding training set to calculate mean squared error (MSE).

At last, only one group of parameters with the minimal MSE value was used as optimum parameters of this SVM model. Thus, we got 3 groups of optimum parameters to develop SVM models.

5.The process of optimizing parameters of Deep Neural Network (DNN) models

DNN is an artificial neural network with more than one intermediate layer and more neurons in each layer. The DNN contains an input layer, an output layer and several hidden layers. A neuron is a computing and storage unit in layers. The output of a neuron is calculated by Equation (5):

$$O = f(\sum_{i=1}^{N} w_i x_i + b) \quad (5)$$

O represents the output. *f(x)* is the activation function for a layer. x_i is the input value or the ouput value of a hidden layer and w_i is the weight of each input or connection between neurons. b is a default bias term. The parameters were chosen by cross-validation. We set the number of hidden layers as 1, 2 and 3. The number of neurons in each hidden layer, varied from 2^1 to 2^{10}.

We chose the sigmoid function as the activation function for DNN. The function is shown in Equation (6):

$$f(x) = \frac{1}{1+e^{-x}} \qquad (6)$$

In the process of training, the loss function was minimized by optimizing the weights and bias of neurons using the backward propagation (BP) algorithm. And in this process, the training set was divided into several batches to optimize the weights and bias randomly. If all batches were used, the training process completed one epoch. The number of batches and epochs are related to the convergence and stability of the model. The number of epoch should be set as large as possible. The network was trained for 2000 epochs with a batch of 100. The optimizer function uses the gradient decent rule for the training process. The optimizer "Adadelta" was used for the models. The learning rate was set as 1.0. The learning rate decay over each update was set as 0.