中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生数学建模竞赛

学　　校	北京石油化工学院
参赛队号	**21100170019**
队员姓名	**1.邢晓龙**
	2.王利猛
	3.张婧

中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生数学建模竞赛

题 目 基于抗乳腺癌候选药物的优化建模分析

摘 要

乳腺癌已成为我国城市女性死亡率最高的恶性肿瘤，其发病率位居女性恶性肿瘤的首位，严重威胁着广大妇女的生命健康。乳腺癌的发展与雌激素受体α亚型（Estrogen receptors alpha, ERα）密切相关。ERα的生物活性值越大，越容易患乳腺癌。IC50 是表征化合物对ERα的生物活性值。IC50 越小，对抑制 ERα活性越有效；而 pIC50（即 IC50 值的负对数）通常与化合物对 ERα生物活性具有正相关性。即 pIC50 越大，对抑制 ERα活性越有效。本文根据文件“Molecular_Descriptor.xlsx”、“ERα_activity.xlsx”以及“ADMET.xlsx”提供的数据，找出对于各个化合物的 pIC50 具有显著影响的分子描述符；构建 pIC50 与 20 个自变量分子描述符之间的逻辑关系并根据给定的自变量预测出 50 个化合物的 pIC50；构建自变量分子描述符对于 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型和性质预测；探究哪些分子描述符的 pIC50 值最大并且 ADMET 效果更好。

问题一：本题题意目的是研究 729 个分子变量对 pIC50 影响的相关度并排序，选取其中影响最显著的 20 个分子描述符。首先对数据集进行观察，注意到数据的分布未知，并且出现离散数据。由于灰色理论对数据的要求比较低，故本问题选取灰色关联度分析方法，计算并提取相关度排序靠前的变量，作为后续问题研究的要素指标。最后筛选出 20 个对ERα的生物活性影响最显著的分子描述符变量，并给出相关度表 5.1。

问题二：由问题一中所得相关度最高的 20 个变量进行后续实验。目的是预测问题二中表 ERα_activity_test 所给出的 50 个化合物 pIC50 值，根据表 Molecular_Descriptor_training 和表 ERα_activity_training 建立 20 个自变量分子描述符相对于 pIC50 的 BP 神经网络模型。再将表 Molecular_Descriptor_test 取出对应列作为输入到模型中求得输出。由于输入变量比较多，且 BP 神经网络的初始权值阈值随机分配容易造成局部最优解，影响实验的准确性。因而采用遗传算法对初始权值阈值进行优化分配。综上，此问采用基于遗传算法的 BP 神经网络模型。经过不断调整种群规模及进化代数等相关参数，得到 50 组 pIC50 的近似最优解，结果已经填入附件“ERα_activity”的 test 表中的 IC50_nM 列及对应的 pIC50 列。

问题三：分析题意可知问题三本质是二分类问题，同时考虑了对问题四的求解，在训练模型时直接使用问题二的训练数据。这时的数据具有非线性、离散型、数据稀疏的特点，综合考虑模型特点最后选取梯度提升决策树模型作为解题模型。模型训练后，对表 Molecular_Descriptor_test 中的数据进行分类预测，将分类得到的结果存于附件“ADMET.xlsx”的 test 表中对应的 Caco-2、CYP3A4、hERG、HOB、MN 列。

为了说明该模型对问题有效性，同时建立线性判别模型，决策树模型，支持向量机，神经网络四个常用分类模型进行训练，使用准确率和运行时间作为评价指标进行对比试验。最后得出该模型对本题的效果最佳的结论。

问题四：本题是一个综合性问题，可以看做是第二问和第三问的结合。本文将此问题分为两步解决。第一步求得第二问模型中，使 pIC50 达到峰值的极值点，也就是 20 个自变量分子描述符的取值。第二步，将此极值点带入第三问训练好的模型内，以求得此组输入参数下的 ADMET 性质。通过不断剔除相关度低的自变量后训练模型，达到同时满足 pIC50 最大和 ADMET 性质匹配度最高（结果与 11010 匹配度最高）为止。最终选取 15 个自变量分子描述符见表 8.1，输出峰值为 6.205，此时 ADMET 性质为 11000 ，与期望性质只存在于性质四的差异。

针对以上问题设计了各种模型的最优参数组合求解算法，并利用 MATLAB 软件编程计算，得到了各个模型在不同条件下的仿真曲线；对结果进行分析，获得了不同分子描述符对 pIC50 的影响规律。总结了模型的优点与不足，为后续研究此类问题的学者提供了一个新的思路，继续优化模型和算法。

关键词：灰色关联；GA-BP 网络；梯度提升决策树；模型优化

目录

一、问题重述

1.1 问题背景

乳腺癌是目前世界上最常见，致死率较高的癌症之一。乳腺癌的发展与雌激素受体密切相关，有研究发现，雌激素受体α亚型（Estrogen receptors alpha, ERα）在不超过 10%的正常乳腺上皮细胞中表达，但大约在 50%-80%的乳腺肿瘤细胞中表达；而对 ERα基因缺失小鼠的实验结果表明，ERα确实在乳腺发育过程中扮演了十分重要的角色。目前，抗激素治疗常用于 ERα表达的乳腺癌患者，其通过调节雌激素受体活性来控制体内雌激素水平。因此，ERα被认为是治疗乳腺癌的重要靶标，能够拮抗 ERα活性的化合物可能是治疗乳腺癌的候选药物。比如，临床治疗乳腺癌的经典药物他莫昔芬和雷诺昔芬就是 ERα拮抗剂。

目前，在药物研发中，为了节约时间和成本，通常采用建立化合物活性预测模型的方法来筛选潜在活性化合物。一个化合物想要成为候选药物，除了需要具备良好的生物活性（此处指抗乳腺癌活性）外，还需要在人体内具备良好的药代动力学性质和安全性，合称为 ADMET（Absorption 吸收、Distribution 分布、Metabolism 代谢、Excretion 排泄、Toxicity 毒性）性质。其中，ADME 主要指化合物的药代动力学性质，描述了化合物在生物体内的浓度随时间变化的规律，T 主要指化合物可能在人体内产生的毒副作用。一个化合物的活性再好，如果其 ADMET 性质不佳，比如很难被人体吸收，或者体内代谢速度太快，或者具有某种毒性，那么其仍然难以成为药物，因而还需要进行 ADMET 性质优化。

1.2 问题提出

问题 1. 根据文件“Molecular_Descriptor.xlsx”和“ERα_activity.xlsx”提供的数据，针对 1974 个化合物的 729 个分子描述符进行变量选择，根据变量对生物活性影响的重要性进行排序，并给出前 20 个对生物活性最具有显著影响的分子描述符（即变量），并请详细说明分子描述符筛选过程及其合理性。

问题 2. 请结合问题 1，选择不超过 20 个分子描述符变量，构建化合物对 ERα生物活性的定量预测模型，请叙述建模过程。然后使用构建的预测模型，对文件“ERα_activity.xlsx”的 test 表中的 50 个化合物进行 IC50 值和对应的 pIC50 值预测，并将结果分别填入其表中的 IC50_nM 列及对应的 pIC50 列。

问题 3. 请利用文件“Molecular_Descriptor.xlsx”提供的 729 个分子描述符，针对文件“ADMET.xlsx”中提供的 1974 个化合物的 ADMET 数据，分别构建化合物的 Caco-2、CYP3A4、hERG、HOB、MN 的分类预测模型，并简要叙述建模过程。然后使用所构建的 5 个分类预测模型，对文件“ADMET.xlsx”的 test 表中的 50 个化合物进行相应的预测，并将结果填入其表中对应的每一列。

问题 4. 寻找并阐述化合物的哪些分子描述符，以及这些分子描述符在什么取值或者处于什么取值范围时，能够使化合物对抑制 ERα具有更好的生物活性，同时具有更好的 ADMET 性质（给定的五个 ADMET 性质中，至少三个性质较好）。

二、问题分析

2.1 问题一

第一问要求对文件附件一：ERα_activity.xlsx 中生物活性数据与附件二：Molecular_Descriptor.xlsx 中 1974 个化合物样本的 729 个分子描述符数据，分析各分子描述符数据对生物活性数据的相关性大小，同时进行排序。最后选取相关性排序靠前的 20 个分子描述符。需要进行的工作主要有以下几点：

（1）数据的处理，将数据处理成更适合相关性分析方法使用的数据。

（2）使用多种相关性模型对数据进行相关性分析。

（3）对生成的相关性排序，进行选取。

2.2 问题二

本题是旨在利用多变量（若干个分子描述符）去预测 pIC50。所以需要构建多变量对 pIC50 的定量预测模型。这可以通过训练 BP 神经网络进行拟合，从而建立二者之间的对应关系。首先，从给定的表 Molecular_Descriptor_training 中选取问题一中确定的 20 个变量作为输入，表 ERα_activity_training 作为输出。每列有 1975 行，视为 1975 个样本。分割为 1800 和 175 两份，前者用于模型训练与验证，后者用于模型测试。然后，采用遗传算法对初始权值阈值进行优化分配，旨在得到近似最优的权值阈值。再进行 BP 神经网络模型训练。最终，选取表 Molecular_Descriptor_test 中的 20 个变量作为神经网络模型的输入，利用模型计算输出 pCI50 的值。用 EXCEL 进行值计算，因为 IC50_nM=-LOG(pIC50*10^(-9))，所以 pIC50=POWER(10,-IC50_nM)/POWER(10,-9)。实验结束，对比使用遗传算法和不使用遗传算法两种方式的 BP 神经网络拟合的相关系数和预测的准确性。

2.3 问题三

在第三问中，要求利用文件“Molecular_Descriptor.xlsx”提供的 729 个分子描述符数据，对文件“ADMET.xlsx”中提供的 1974 个化合物的 ADMET 数据中 Caco-2、CYP3A4、hERG、HOB、MN 分别建立一个二分类预测模型，简要说明模型的建立。这就是需要我们建立五个二分类预测模型，并且简要叙述模型。需要我们完成的问题是如何进行模型的选取，然后是如何建立模型，以及最后如进行模型的评价。

2.4 问题四

将问题四分为两步进行逐步解决：

（1）对 20 个自变量进行问题二的神经网络训练，训练好的神经网络视为一个函数。通过调用 Matlab 的 fmincon 函数计算有条件极值。约束条件 s.t. 为给定的 Molecular_Descriptor_training 里选取的自变量的最大最小值。编写脚本 extremum 求得本网络的极值 X_1-X_{20} 以及最大值 Z_{max}

（2）求得的极值点 X_1-X_{20} 带入问题三的模型中，输出 ADMET 性质

如果 ADMET 性质与期望性质 11010 匹配度较小，则剔除相关度较低的变量（本文选择第一次训练 20 个，第二次 15 个，第三次 10 个，第四次 8 个）重复训练，综合多次训练结果输出的生物活性和 ADMET 性质进行选择

三、模型假设

1.问题一假设：在模型使用时，需要对学习器产生的残差进行累加，在调参数优化增加学习器，运行时间也会增加。同时学习器之间存在依赖关系，难以并行训练数据。

2.问题二假设：题目附表所给定数据均为有效数据，不需要进行数据清洗。

3. 问题四假设：期望的 ADMET 性质为 11010，即该化合物的小肠上皮细胞渗透性较好，能够被 CYP3A4 代谢，不具有心脏毒性；口服生物利用度较好且不具有遗传毒性。

四、符号说明

本文所使用的符号系统及其解释如表 4.1 所示。

表 4.1 本文所使用的部分符号说明

符号	符号说明
Y	参考数列
X_i	比较数列
$X_{rescaled}$	无量纲化
$\xi_i(k)$	灰色关联系数
y_i	预测输出
o_i	期望输出
E	误差绝对值和
F	个体适应度
p_i	选择概率
i	个体
F_i	个体的适应度值
N	种群个体数目
a_k	第 k 个染色体
a_{kj}	第 k 个染色体 a_k 在 j 位的交叉操作
a_{1j}	第 1 个染色体 a_1 在 j 位的交叉操作
a_{max}	基因 a_{ij} 的上界
a_{min}	基因 a_{ij} 的下界
y_i	预测值
$\hat{y}_i$	期望值
T	最大迭代次数
L	损失函数
c_{tj}	最佳负梯度拟合值

注：考虑到全文连续性，其他未在表 4.1 中列出的符号将在建模和求解过程中给出解释说明。

五、问题一模型的建立与求解

5.1 数据处理

由于附件一的原始数据都是根据简化分子线性输入规范（SMILES）对分子结构进行计算得到的，因此无需再对异常参数、缺失值进行处理。同时对附件二数据观察，发现有很多全零的列向量。对于生物活性数据，全零列向量代表的分子描述符相关性影响为零，为了提升后来算法的训练效率，对全零列进行清除得到，个分子描述符。

5.2 相关性计算和显著分子描述符选取

最常用的相关性分析方法是线性相关性分析。当一个变量中的变化与另一个变量中的成比例变化相关时，这两个变量具有线性关系。线性相关系数评估两个连续变量之间的线性关系。例如 Pearson 相关。但是所给数据中具有离散值，大量数据信息缺失，数据分布规律未知。因此不满足线性相关性分析方法的条件。

灰度关联度算法的灰色关联分析对样本量的多少和样本有无明显的规律要求较低，且计算量小。因此灰色关联分析模型可以解决所给数据中具有离散值，大量数据信息缺失，数据分布规律未知的情况所带来的问题[1]。

由上述分析，选用灰色关联分析进行相关性分析，灰色关联分析是指对一个系统发展变化态势的定量描述和比较的方法，其基本思想是通过确定参考数据列和若干个比较数据列的几何形状相似程度来判断其联系是否紧密，它反映了曲线间的关联程度。通常可以运用此方法来分析各个因素对于结果的影响程度，也可以运用此方法解决随时间变化的综合评价类问题，其核心是按照一定规则确立随时间变化的母序列，把各个评估对象随时间的变化作为子序列，求各个子序列与母序列的相关程度，依照相关性大小得出结论。具体计算步骤如下[2]：

（1）确定分析数列

确定反映生物活性（IC50）和影响生物活性的分子描述符的数列。反映生物活性（IC50）特征的数据序列，称为参考数列（Y）。影响生物活性（IC50）的分子描述符组成的数据序列，称比较数列（X_i）。

① 参考数列（又称母序列）

$$Y=\{Y(\mathrm{k})|k=1,2,...,n\} \tag{1}$$

② 比较数列（又称子序列）

$$X_{\mathrm{i}}=\{X_{\mathrm{i}}(\mathrm{k})|\mathrm{k}=1,2,\ldots,\mathrm{n};\mathrm{i}=1,2,\ldots,\mathrm{m}\} \tag{2}$$

(2)无量纲化

由于系统中各因素列中的数据可能因量纲不同，不便于比较或在比较时难以得到正确的结论。因此在进行灰色关联度分析时，一般都要进行数据的无量纲化处理。运用的公式如下:

$$X_{\text{rescaled}}=a+\left[\frac{X-\min_X}{\max_X-\min_X}\right](\mathrm{b}-\mathrm{a}) \tag{3}$$

公式可以将数据 X 重新缩放到任意区间[a,b]，其中选取 a=0，b=1。X_i 表示第 i 列数

据。

(3)灰色关联系数计算

利用已经确定反映生物活性（IC50）特征的参考数列(Y)和影响生物活性的分子描述符组成的比较数列 X_i（k），进行灰色关联系数计算。灰色关联系数公式如下：

$$\xi_i(k)=\frac{\min\limits_i\min\limits_k|y(k)-x_i(k)|+p\max\limits_i\max\limits_k|y(k)-x_i(k)|}{|y(k)-x_i(k)|+p\max\limits_i\max\limits_k|y(k)-x_i(k)|} \tag{4}$$

$\rho\in(0，\infty)$，称为分辨系数。ρ越小，分辨力越大，一般ρ的取值区间为$(0,1)$具体取值可视情况而定。通常取ρ = 0.5。

5.3 相关性计算结果和综合选择

选用灰色关联分析方法，运用 MATLAB 对灰色关联系数进行程序编写和运行，对原数据集求出相关性，并进行排序对前二十个进行选取。最终结果如表 5.1 所示。

表 5.1 主要变量筛选结果

变量名称	符号表示	相关度
LipoaffinityIndex	X_1	0.803368
MDEC-23	X_2	0.799675
SwHBa	X_3	0.793004
MLogP	X_4	0.788023
n6Ring	X_5	0.786119
nwHBa	X_6	0.786011
nRing	X_7	0.785649
C2SP2	X_8	0.783767
ETA_Psi_1	X_9	0.781164
nT6Ring	X_{10}	0.780885
BCUTp-1h	X_{11}	0.779689
minsBr	X_{12}	0.779192
naAromAtom	X_{13}	0.776696
mindssC	X_{14}	0.775814
WTPT-2	X_{15}	0.771931
nAromBond	X_{16}	0.770420
nBondsM	X_{17}	0.769229
minwHBa	X_{18}	0.768429
ETA_Epsilon_3	X_{19}	0.768250
SsBr	X_{20}	0.768224

六、问题二模型的建立与求解

6.1 算法简介

遗传算法优化 BP 神经网络分为 BP 神经网络结构确定、遗传算法优化和 BP 神经网络预测 3 个部分。其中，BP 神经网络结构确定部分根据拟合函数输入输出参数个数确定,这样就可以确定遗传算法的优化参数个数，进而确定遗传算法个体的编码长度。因为遗传算法优化的参数是 BP 神经网络的初始权值和阈值，只要网络的结构已知，权值和阈值的个数就已知了。

6.1.1 BP 神经网络

BP 神经网络处理信息的单元一般分为三类：输入单元、输出单元和隐含单元[3]。 顾名思义：输入单元接受外部给的信号与数据；输出单元实现系统处理结果的输出；隐含单元处在输入和输出单元之间，从网络系统外部是无法观测到隐含单元的结构的。除了上述三个处理信息的单元之外，神经元间的连接强度大小由权值等参数来决定。

6.1.2 遗传算法

神经网络可以理解为异常复杂以至于无法套用一般化公式的函数。它由神经网络结构和权值阈值确定。引入遗传算法就是为了得到神经网络的最优初始权值阈值[4]。

假设权值阈值所对应的所有解是种群，而种群中的个体就是一组权值阈值。遗传算法的工作就是让这个种群中的个体去不断繁衍，在繁衍的过程中一方面会发生染色体交叉而产生新的个体。另一方面，基因变异也会有概率会发生并产生新的个体。接下来，通过计算个体的适应度，根据选择算子淘汰质量差的个体，保留质量好的个体，并且让这个繁衍的过程持续下去，那么最后就有可能进化出最优或者较优的个体。

以下是遗传算法流程及原理规定：

（1）种群初始化

个体编码方法为实数编码，每个个体均为一个长度为 903 的实数向量，由输入层与隐含层连接权值、隐含层阈值、隐含层与输出层连接权值以及输出层阈值 4 部分组成。个体包含了神经网络全部权值和阐值，在网络结构已知的情况下，就可以构成一个确定的神经网络。

（2）适应度函数

把预测输出 y_i 和期望输出 o_i 之间的误差绝对值和 E 作为个体适应度 F，计算公式为

$$F=k\left(\sum_{i=1}^{n} abs\left(y_i-o_i\right)\right) \tag{5}$$

k 为系数

（3）选择操作

轮盘赌：基于适应度比例的选择策略，每个个体 i 的选择概率 p_i 为

$$f_i=k/F_i$$

$$p_i=\frac{f_i}{\sum_{j=1}^{N} f_i} \tag{6}$$

式中，F_i 为个体 i 的适应度值，由于适应度值越小越好，所以在个体选择前对适应度值取倒数，k 为系数，N 为种群个体数目。

（4）交叉操作

由于个体采用实数编码，所以交叉操作方法采用实数交叉法，第 k 个染色体 a_k 和第 1 个染色体 a_l 在 j 位的交叉操作方法如下：

$$\begin{aligned} a_{kj}&=a_{kj}(1-b)+a_{1j}b \\ a_{1j}&=a_{1j}(1-b)+a_{kj}b \end{aligned} \tag{7}$$

式中，b 是[0, 1]之间的随机数。

（5）变异操作

选取第 i 个个体的第 j 个基因进行变异，变异操作方法如下：

$$a_{ij}=\begin{cases} a_{ij}+(a_{ij}-a_{\max})*f(g)r\geq 0.5 \\ a_{ij}+(a_{\min}-a_{ij})*f(g)r\leq 0.5 \end{cases} \tag{8}$$

式中，a_{max}、a_{min} 为基因 a_{ij} 的上界和下界，$f(g)=r_2(1-g/G_{max})^2$，r_2 为一个随机数，g 为当前迭代次数，G_{max} 为最大进化次数，r 为[0,1]之间的随机数。

6.2 模型的建立

6.2.1 BP 神经网络结构确定

本题选用三层神经网络。n_1，n_2 分别代表输入层与隐藏层神经元个数。其中问题二样本有 20 个输入参数，1 个输出参数，所以这里 $n_1=20$，由于隐藏层神经元个数与输入层神经元个数有以下的近似公式：

$$n_2=2*n_1+1 \tag{9}$$

故 $n_2=41$。

BP 神经网络结构：20-41-1 即输入层、隐藏层、输出层的节点数分别为 20 个、41 个、1 个。

权值总数：20*41+41*1=861 个

阈值总数：41+1=42 个

遗传算法优化参数个数：861+42=903 个

BP 神经网络结构如图 6.1 所示。其中 X_1-X_{20} 代表问题一选定的 20 个自变量分子描述符，Y 代表因变量 pIC50。

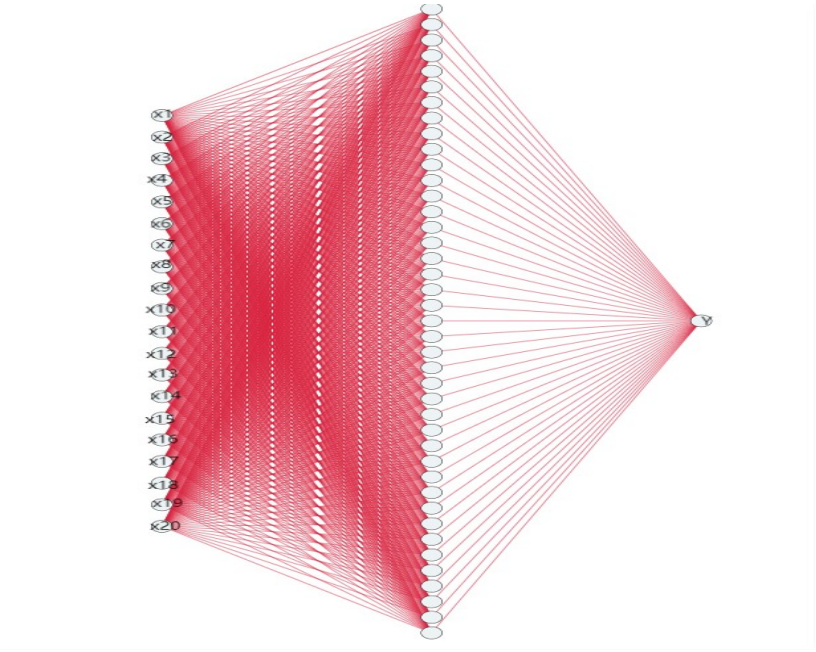

图 6.1 BP 神经网络结构图

6.2.2 遗传算法运行参数设定

表 6.2 遗传算法运行参数

种群大小	进化代数	交叉概率	变异概率
40	15	0.5	0.05

这里的参数选取是根据重复进行试参确定。一般来说，交叉概率应在 0.3-0.9，变异概率在 0.01-0.1。交叉概率和变异概率是指随机种群有多大可能会发生这些操作。在重复性实验中，算法最多在迭代 15 次之后，适应度函数已经收敛，为了缩短计算时间，算法里把进化代数设定为 15。种群大小即个体数太少在逐代进化的过程中不容易产生更优的染色体，太多则计算时间过长使得种群规模过于庞大。因此种群大小一般不超过 40。本题采取进化到 15 代后停止进化，输出当前最优染色体，即代表最优权值阈值的实数向量信息。

6.3 算法实现

本研究的问题属于优化问题，进而构建优化模型。遗传算法在求解多变量优化问题具有较好的适用性，嵌入神经网络模型能更好的求解此问题，因此本研究拟通过遗传算法优化 BP 神经网络的算法求解模型。算法总体流程如图 6.3 所示[5]。

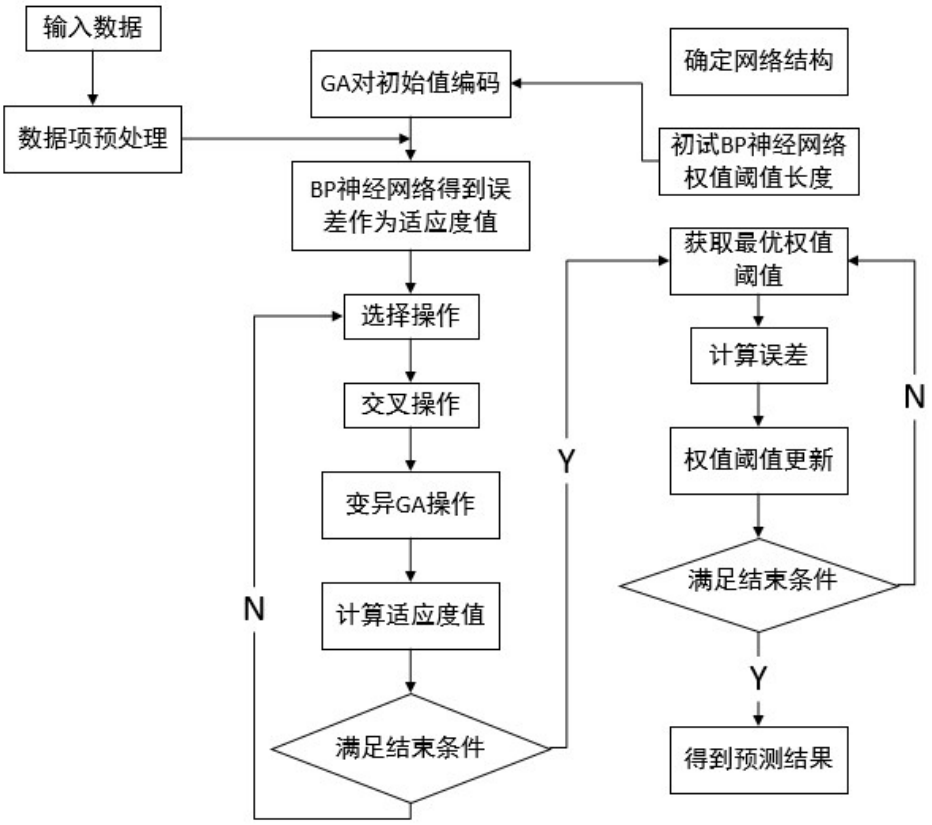

图 6.3 遗传算法优化 BP 神经网络的算法流程

种群中的每个个体都包含了一个网络所有权值和阈值,个体通过适应度函数计算个体适应度值,遗传算法通过选择、交叉和变异操作找到最优适应度值对应的个体。

BP 神经网络预测用遗传算法得到最优个体对网络进行初始权值和阈值的赋值,网络经训练后预测样本输出。

神经网络的权值和阈值一般是通过初始化为[-0.5,0.5]区间的随机数，这个初始化参数对网络训练的影响很大，但是又无法准确获得，对于相同的初始权重值和阈值，网络的训练结果是一样的，引入遗传算法就是为了优化出最优的初始权值和阈值。

现对以上算法流程进行如下简要说明：

（1）导入数据，将训练数据与测试数据归一化。确定 BP 神经网络结构，为 20-41-1，所以需要优化的参数总数为 20*41+41*1=903，即为染色体长度。

（2）对一个种群所有个体进行初始化，即对每个个体的染色体编码。产生个数为种群大小的一组染色体，染色体存储结构为 903 维的实数向量。分别计算它们的适应度。并记录下初始种群的平均适应度和最优染色体。

（3）开始进化，对父代进行选择，父代到子代进行交叉，变异操作。解码计算子代每个个体的适应度，将子代适应度最小的个体染色体与父辈的最优染色体适应度进行比较，若小于父辈则替代成为最优染色体，大于则不变。同样记录该代的平均适应度和最优染色体。

（4）继续进化，直到达到预设进化代数，输出最优染色体。即为 BP 网络最优初始权值阈值组合。

（5）将最优权值阈值分配给 BP 网络，对输入数据和输出数据进行训练。即进行计算误差,权值更新的有限次循环。程序设定最大循环次数 100,学习率 0.1,期望误差 0.00001。达到最大循环次数或误差小于期望误差时结束训练。此时的模型即为最优 BP 模型。

6.4 结果分析

由于 BP 网络随机分配权值阈值的不确定性，导致每次生成的模型都不一致。此外，虽然使用遗传算法寻得的是最优染色体，但是只是相对于每次运行程序而言，不同次的训练也会使得生成的最优染色体不一致。所以每次的实验的结果都不一样。本题对于未使用和使用遗传算法优化的 BP 神经网络分别进行多次模型训练。最终采取每种方式相对具有代表性的两次实验结果，在下文进行数据对比分析。

计算值说明：

（1）MAE，平均绝对值误差，它表示预测值 y_i 和期望值 $\hat{y}_i$ 之间绝对误差（残差）的平均值。做非线性拟合时，MAE 越小越好。

$$\frac{1}{\mathrm{m}}\sum_{i=1}^{m}\left|(y_i-\hat{y}_i)\right| \tag{10}$$

（2）RMSE，均方根误差，它表示预测值 y_i 和期望值 $\hat{y}_i$ 之间绝对误差（残差）的样本标准差。是用来衡量预测值同期望值之间偏离的离散程度。预测值同期望值偏离的程度越大，RMSE 越大。做非线性拟合时，RMSE 越小越好。

$$\sqrt{\frac{1}{\mathrm{m}}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2} \tag{11}$$

（3）MSE，均方误差，为 RMSE 的平方值。

$$\frac{1}{\mathrm{m}}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2 \tag{12}$$

m 为样本数

6.4.1 未用遗传算法优化的 BP 网络模型

表 6.4 第一次与第二次未用遗传算法优化的 BP 网络模型误差计算

	第一次 BP	第二次 BP
平均绝对误差 MAE	1.3688	1.3703
均方根误差 RMSE	1.6079	1.636
均方误差 MSE	2.5852	2.6766

测试集预测值与期望值对比图示：

红色星号代表训练的网络对输入测试集的预测数据，蓝色圆圈代表测试集对应输出的实际值（期望值），蓝色方块代表二者误差。如图 6.5 和图 6.6 所示，大部分误差在[-4,4]

区间内进行波动，有一部分误差已经大于 4，从而造成预测值与期望值之间拟合度很低。

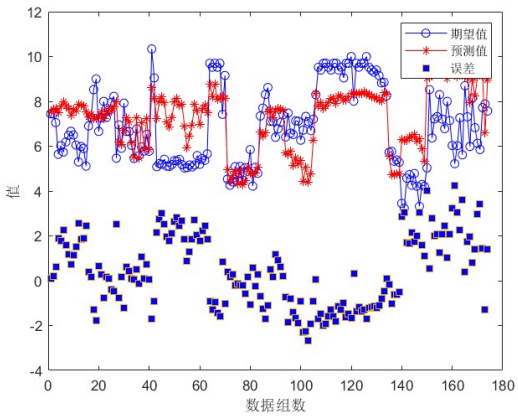

图 6.5 第一次期望值与预测值对比图

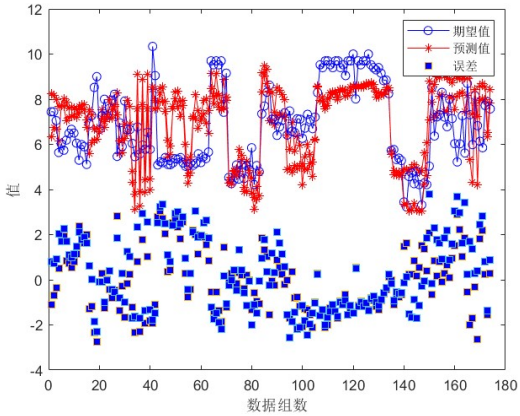

图 6.6 第二次期望值与预测值对比图

误差图 6.7 和图 6.8 所示：

第一次在迭代第 14 次获得最小误差为 0.0125，第二次在迭代第 10 次获得最小误差为 1.0146

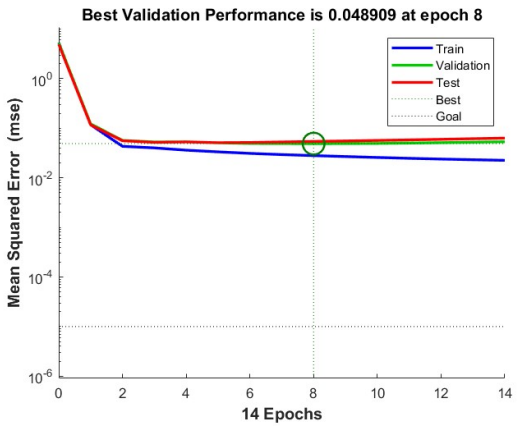

图 6.7 第一次误差变换图

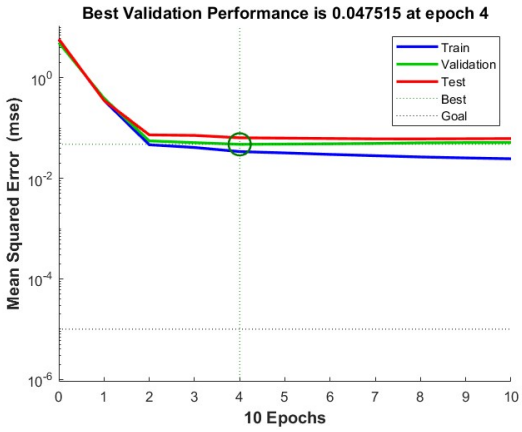

图 6.8 第二次误差变换图

训练状况图 6.9 和图 6.10 所示:

两次的梯度都是下降-上升-下降-上升-下降的走向。没有达到六次不升反降。

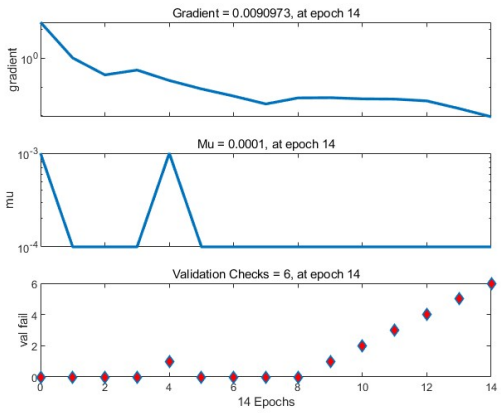

图 6.9 第一次梯度变换图

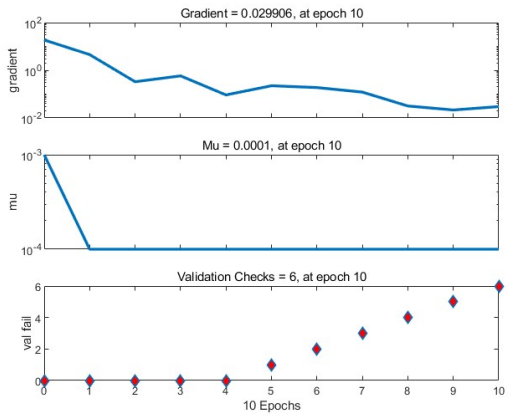

图 6.10 第二次梯度变换图

回归曲线图 6.11 和图 6.12 所示：

两次实验的训练，验证，测试数据集拟合度依次下降。第一次实验总体拟合系数为 0.86605，第二次实验总体拟合系数为 0.84311。

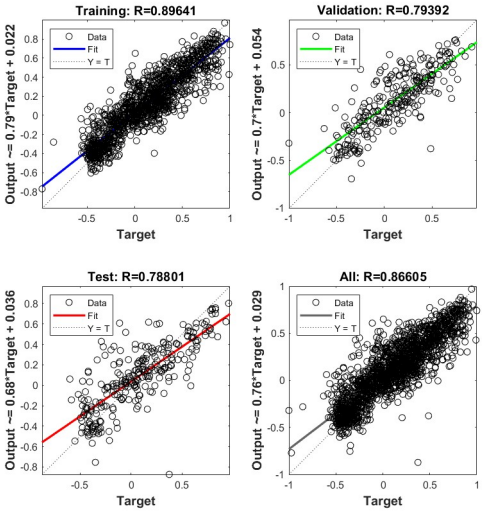

图 6.1 第一次数据集总体回归图

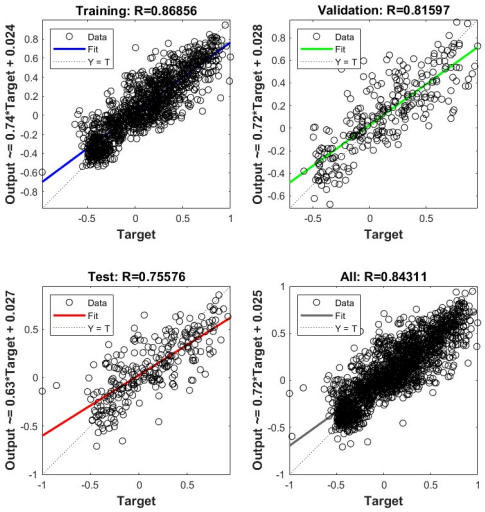

图 6.12 第二次数据集总体回归图

6.4.2 使用遗传算法优化的 BP 网络模型

表 6.13 第一次与第二次使用遗传算法优化的 BP 网络模型误差计算

	第一次 GA-BP	第二次 GA-BP
平均绝对误差 MAE	1.1907	1.1331
均方根误差 RMSE	1.4334	1.3432
均方误差 MSE	2.0545	1.8041

适应度曲线图 6.14 和图 6.15 所示：

第一次进化 13 代后种群适应度函数收敛，第二次进化 14 代后种群适应度函数收敛。

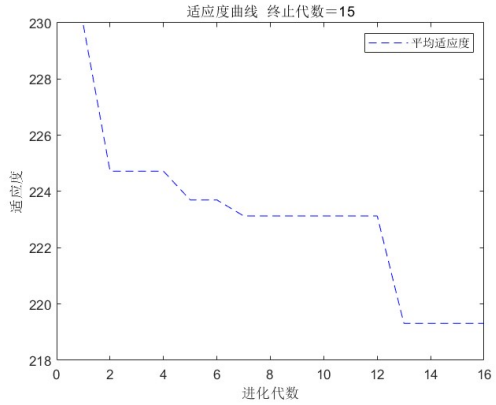

图 6.14 第一次种群进化适应度曲线变化图

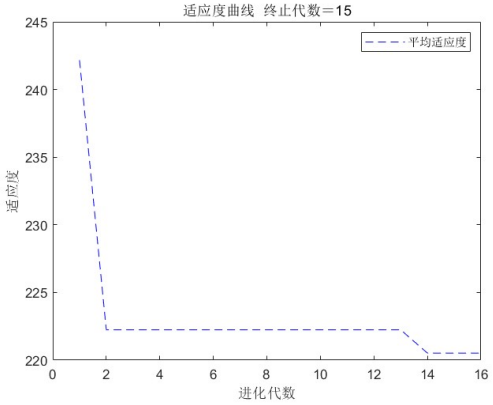

图 6.15 第二次种群进化适应度曲线变化图

测试集预测值与期望值对比图 6.16 和图 6.17 所示：

所有误差都在[-4, 4]区间内进行波动，相较于未用遗传算法优化的 BP 网络，误差波

动比较小。使得预测值与期望值之间拟合度比较高。

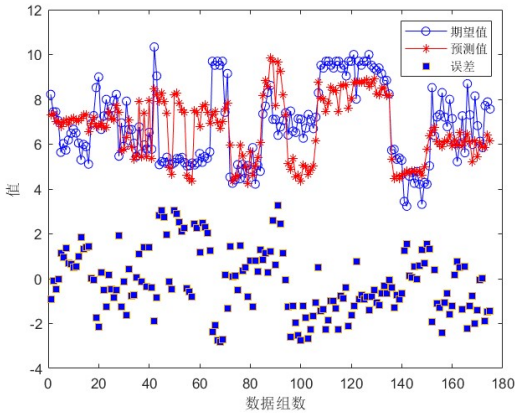

图 6.16 第一次期望值与预测值对比图

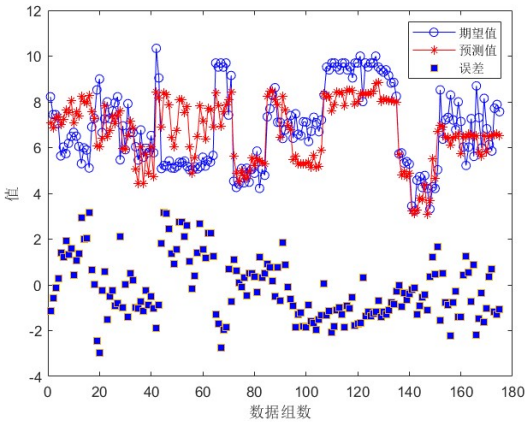

图 6.17 第二次期望值与预测值对比图

误差图 6.18 和图 6.19 所示：

第一次在迭代第 13 次获得最小误差为 0.0186，第二次在迭代第 10 次获得最小误差为 0.0052。

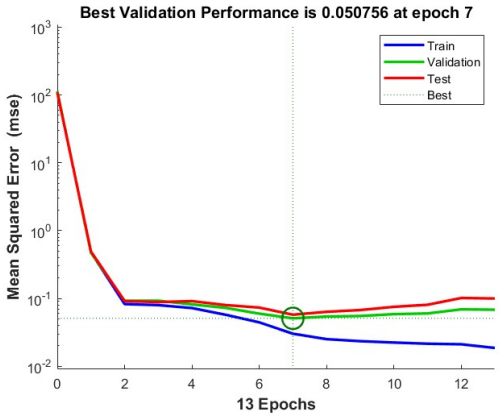

图 6.18 第一次误差变换图

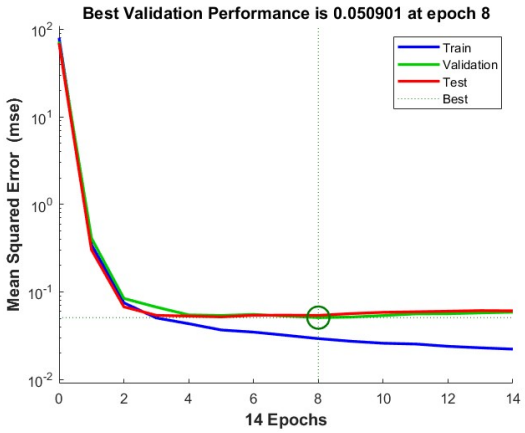

图 6.19 第二次误差变换图

训练状况图 6.20 和图 6.21 所示:

两次的梯度都是下降-上升-下降-上升-下降的走向。没有达到六次不升反降。

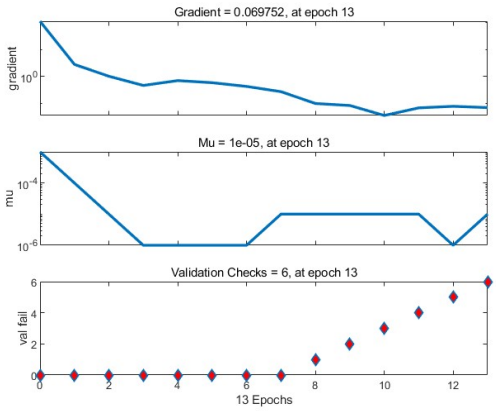

图 6.20 第一次梯度变换图

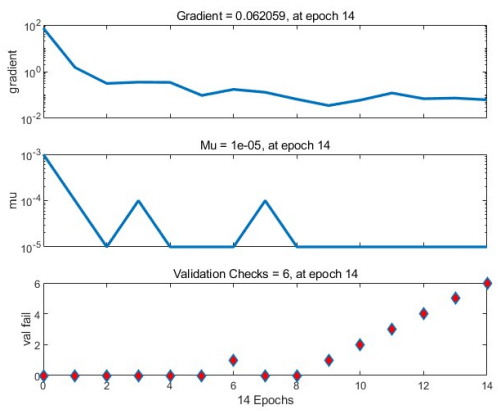

图 6.21 第二次梯度变换图

回归曲线图 6.21 和图 6.22 所示：

两次实验的训练，验证，测试数据集拟合度差别不大，两两之间未超过 0.1。第一次实验总体拟合系数为 0.86259，第二次实验总体拟合系数为 0.86147。

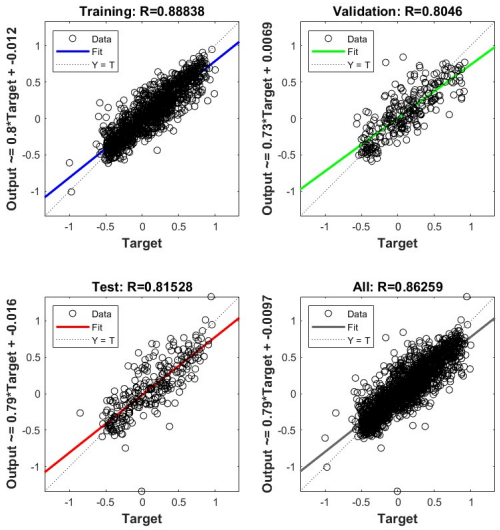

图 6.21 第一次数据集总体回归图

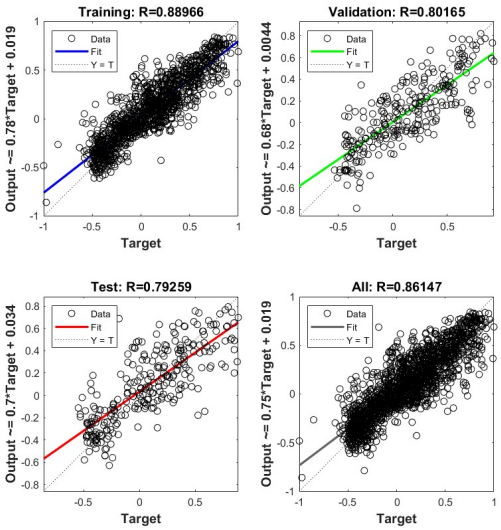

图 6.22 第二次数据集总体回归图

对两种模型的总结：

总体而言，有无使用遗传算法对神经网络进行优化对整体数据集的预测改变不大。但是，使用遗传算法的模型对测试集的拟合程度（拟合系数分别为 0.81528，0.86259）比未用遗传算法的模型的测试集的拟合程度（拟合系数分别为 0.78801，0.0.75576）要普遍更高。这也就说明了，对同一组测试集而言，使用遗传算法对神经网络进行初始优化会使得预测结果更为靠近期望值。

除此之外，未用遗传算法优化的模型其 MAE（1.3688,1.3703）、RMSE（1.6079,1.636）也普遍高于用遗传算法的模型的 MAE（1.1907,1.1331）、RMSE（1.4334，1.3432）也就是说，后者的数据更贴近于期望值且波动不大，更为稳定。

七、问题三模型的建立与求解

7.1 模型的选取

7.1.1 线性与非线性模型选择

因为药物分子的结构复杂性，分子结构作用的复杂性，导致分子描述符数据大概率对生物活性的影响是非线性的。这样我们选用线性的分类方法（逻辑回归）来建立模型，很难得到很好的分类效果。因此我们应该选用非线性的分类（例如决策树）来建立我们的分类模型。

7.1.2 传统分类模型与深度神经网络模型

为了方便第四问的模型结合问题，第三问的数据直接以第二问训练所用的分子描述符数据。由于只有 20 个分子描述符数据，数据集规模大大减少，但是深度神经网络的训练需要大量的数据，同时所需要的资源很大。如果数据过少，神经网络模型还会出现过拟合的问题。对于这种小规模数据的分类问题传统分类模型缺可以做的很好。根据奥卡姆剃刀原则，两个分类模型在相差不大的分类效果下，选择结构简单的模型更好一些。因此我们选用传统分类模型，进行训练。其中提升树是传统模型中的一个典型模型之一，并且可以训练出很好的分类效果。

7.2 基于梯度提升决策树分类模型的建立

梯度提升决策树（Gradient Boosting Decision Tree，GBDT）是以决策树为基学习器的一种 Boosting 算法，它在每一轮迭代中建立一个决策树，使当前模型的残差在梯度方向上减少[6]；然后将该决策树与当前模型进行线性组合得到新模型；不断重复，直到决策树数目达到指定的值，得到最终的强学习器。梯度提升决策树算法主要优点是可以灵活处理各种类型的数据，包括连续值和离散值。同时在相对少的调参时间情况下，预测的准确率也可以比较高。

在确定要使用的分类模型后，主要使用三个步骤对模型建立，模型的求解、模型的验证：

（1）从问题一筛选出的结果数据进行再处理，主要工作是：训练数据的准备，数据集划分。

（2）利用训练集数据对基于梯度提升决策树（GBDT）模型的 ADMET 数据（Caco-2、CYP3A4、hERG、HOB、MN）五个分类模型分别进行训练。最后附件所给测试集数据进行分类，得到对应的预测分类结果。

（3）另外使用四种分类模型与梯度提升决策树模型进行对比实验，得到所有分类结果以后，使用验证准确率和测试准确率和运行时间作为评价指标对结果进行综合分析论证。

7.2.1 数据处理

由于使用的第一问的结果数据，数据已经经过了基本处理。因此无需对数据进行异常值和缺失值的再次处理。主要进行训练集的准备和数据集的划分。

（1）训练集的准备

（2）由于根据 ADMET 数据训练五个模型，所以我们要建立五个相应的训练集。

（3）数据集的划分

（4）我们将准备好的数据集的样本数据，划分 15%的样本数据仅作为模型测试集，用来评估泛化能力。

7.2.2 模型算法的建立

模型输入的是处理好的样本训练集 T={（x，y_1）,(x_2，y_2),...(x_m，y_m)}，使用最大迭代次数 t, 损失函数 L。最终输出为强学习器 F（x）。

初始化弱学习器

$$\mathrm{f}_0(x)=\underbrace{\arg\min}_{c}\sum_{i=1}^{m}L(y_i,c) \tag{13}$$

对于二元 GBDT，如果用类似于逻辑回归的对数似然损失函数，则损失函数为：

$$L(\mathrm{y},\mathrm{f}(\mathrm{x}))=\log(1+\exp(-\mathrm{y}\mathrm{f}(\mathrm{x}))) \tag{14}$$

对迭代轮数 t=1,2,...T 有：对样本 i=1,2，...m

其中 y∈{-1, 1}则此时的负梯度误差为

$$r_{ti}=-\left[\frac{\partial L(\mathrm{y},\mathrm{f}(\mathrm{x_i}))}{\partial f(x_i)}\right]_{f(x)=f_t-1(x)}=y_i/(1+\exp(y_i f(x_i))) \tag{15}$$

对于生成决策树，我们各个叶子节点的最佳负梯度拟合值为

$$\mathrm{c}_{\mathrm{tj}}=\underbrace{\arg\min}_{c}\sum_{x_i\in R_{tj}}\log(1+\exp(-y_i(f_t\text{-}\mathrm{l}(x_i)+c))) \tag{16}$$

由于上式比较难优化，我们一般使用近似值代替

$$\mathrm{c}_{\mathrm{tj}}=\sum_{x_i\in R_{tj}}r_{ti}\Big/\sum_{x_i\in R_{tj}}|r_{ti}|(1-|r_{ti}|) \tag{17}$$

更新强学习器

$$\mathrm{f}_{\mathrm{t}}(x)=\mathrm{f}_{\mathrm{t\text{-}1}}(x)\sum_{j=1}^{J}c_{tj}I(x\in R_{tj}) \tag{18}$$

得到强学习器 $f(x)$ 的表达式

$$\mathrm{f}(x)=\mathrm{f}_{\mathrm{T}}(\mathrm{x})=\mathrm{f}_0(x)+\sum_{t=1}^{T}\sum_{j=1}^{J}c_{tj}I(x\in R_{tj}) \tag{19}$$

得到了预测函数 $F_M(x)$，用来进行概率估计：

$$P(y=1|x)=p=\frac{e^{2F(x)}}{1+e^{2F(x)}}=\frac{1}{1+e^{-2F(x)}} \tag{20}$$

$$P(y=-1|x)=1-p=\frac{1}{1+e^{2F(x)}} \tag{21}$$

最后我们使用的得到的概率，对样本进行分类。

7.3 模型训练和测试

将训练集数据放入搭建好的模型中，同时因为使用训练集数据量小，所以采用 5 折的交叉验证法，对分类模型进行训练和验证。最后得到训练后的模型，验证准确率以及运行时间。然后将相分离出来的测试集放到模型中进行测试，获得测试准确率。根据 ADMET.xlsx 文件更改相应变量，得到五个不同的训练数据集，并训练出相对应的模型。由于篇幅限制，方便其他的分类方法进行联合分析，在下面对各个方法模型的分析中给出具体的验证数据和过程。

7.4 实验结果

在模型训练好后，将 Molecular_Descriptor.xlsx 文件中 test 表中的数据，根据第一问的筛选出的 20 个分子描述符进行相应挑选，生成一个分类预测数据集。将数据集导入到训练好的五个分类模型中，分别生成一个 50*1 的向量，最终获得一个 50*5 的结果矩阵也就是分类完成后的数值矩阵，其中值为 0 或 1 将生成的 50*5 的矩阵保存到附件 3.1 中。

7.5 模型验证和对比实验

7.5.1 分类模型介绍

在使用梯度提升决策树（GBDT）模型的同时，我们也注意到其他机器学习的模型，也可以用来做二分类的模型训练。为了对该模型的有效性，有一个清楚的认识。我们选取了 4 个常见的机器学习分类模型，对梯度提升决策树模型进行比较分析。我们选用的算法有：线性判别模型，决策树模型，支持向量机，神经网络。

（1）线性判别模型（LDA）[7]：LDA(Linear Discriminant Analysis)是一种经典的线性判别方法，又称 Fisher 判别分析。该方法思想比较简单:给定训练集样例，设法将样例投影到一维的直线上，使得同类样例的投影点尽可能接近和密集（即希望类内离散度越小越好），异类投影点尽可能远离（即希望两类的均值点之差越小越好）

（2）K 最近邻分类器（KNN）[8]：KNN 主要是对距离来进行分类，距离的取值取决于不同特征值的选取。在空间里一个点，它的 k 个最相近的点所属的种类，大概率跟该点是一样的。这样样本最后也被归结为该种类。

（3）决策树（DT）[9]：决策树是一种特殊的树形结构，一般由节点和有向边组成。其中，节点表示特征、属性或者一个类。而有向边包含判断条件。决策树从根节点开始延伸，经过不同的判断条件后，到达不同的子节点。而上层子节点又可以作为父节点被进一步划分为下层子节点。一般情况下，我们从根节点输入数据，经过多次判断后，这些数据就会

被分为不同的类别。这就构成了一颗简单的分类决策树。

（4）支持向量机（SVM）[10]：支持向量机(SVM)的全称是 Support Vector Machine，其含义是支持向量运算的分类器，主要用于解决模式识别领域中的数据分类问题，属于有监督学习算法的一种。其中“机”的意思是机器，可以理解为分类器。在求解的过程中，会发现只根据部分数据就可以确定分类器，这些部分数据称为支持向量。支持向量机要解决的问题可以用一个经典的二分类问题加以描述。

7.5.2 评价指标

（1）准确率

准确率（accuracy）计算公式如下所示：

$$\text{accuercy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{alldate} \tag{22}$$

准确率表示预测正确的样本（TP 和 TN）在所有样本（alldata）中占的比例。在数据集不平衡时，准确率将不能很好地表示模型的性能。可能会存在准确率很高，而少数类样本全分错的情况，此时应选择其它模型评价指标。由于加入到模型的数据类型不懂，分为验证准确率和测试准确率。

（2）运行时间

运行时间就是，相应模型，从训练集加入到模型中从开始训练到训练结束的时间。

7.5.3 模型参数说明

为了模型的对比方法，仅使用 ADMET 数据集中 Caco-2 列作为训练的响应变量。同时为了保证产生客观的对比结果，所有的模型都仅用默认初始值的模型进行训练。如表 6.1 显示是各个模型的参数：

表 6.1 各个模型的参数

方法名称	参数名称	参数
线性判别（LDA）	DiscrimType	linear
	Gamma	0
	FillCoeffs	off
决策树模型（DT）	SplitCriterion	gdi
	MaxNumSplits	100
	Surrogate	off
支持向量机（SVM）	KernelFunction	linear
	KernelScale	auto
	BoxConstraint	1
	Standardize	TRUE
梯度提升决策树（GBDT）	Method	AdaBoostM1
	NumLearningCycles	30

	Learners	template
	LearnRate	0.1
K 最近邻分类器(KNN)	Distance	Euclidean
	NumNeighbors	10
	DistanceWeight	Equal
	Standardize	TRUE

7.5.4 实验分析和展示

按照评价指标来分析，模型的评价指标中验证准确率和测试准确率的值是越大越好，运行时间是越小越好。五个模型的相关评价指标数据，如表 7.2 和图 7.3、7.4、7.5 所示。下面根据准确率和运行时间指标，对模型进行简要分析：

（1）在准确率方面最差的是线性判别（LDA）线性判别模型和支持向量机（SVM）模型。从实验结果，可以有理支持对于线性模型不适用于解决本问题的结论。

（2）其中最优秀的是梯度提升决策树（GBDT）模型，从图表中可以明显看出该模型对于其他一遍机器学习模型的优势。

（3）从运行时间来看，最优秀的是 K 最近邻分类器（KNN）模型，最差为线性判别（LDA）线性判别模型和支持向量机（SVM）。梯度提升决策树（GBDT）模型运行时间适中。

（4）由于对是医药实际情况的需求，显然分类模型的准确度要求优先级更高。所以综合考虑，实验进一步验证了梯度提升决策树（GBDT）模型应用于本问题的有效性。

表 7.2 五个模型的相关评价指标数据

分类方法	验证准确率	测试准确率	运行时间（s）
线性判别（LDA）	0.809	0.884	3.5903
支持向量机（SVM）	0.813	0.866	3.4458
决策树（DT）	0.82	0.88.4	1.5999
K 最近邻分类器（KNN）	0.822	0.87	1.3489
梯度提升决策树（GBDT）	0.848	0.942	2.4761

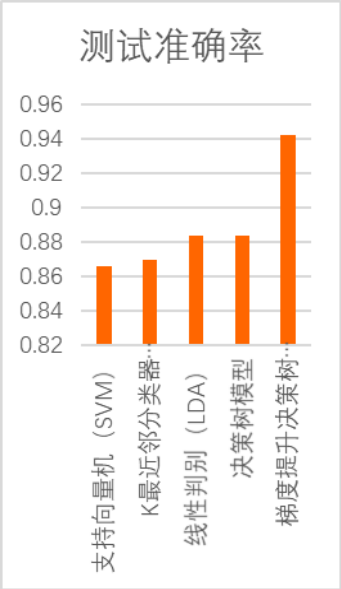

图 7.3 测试准确率

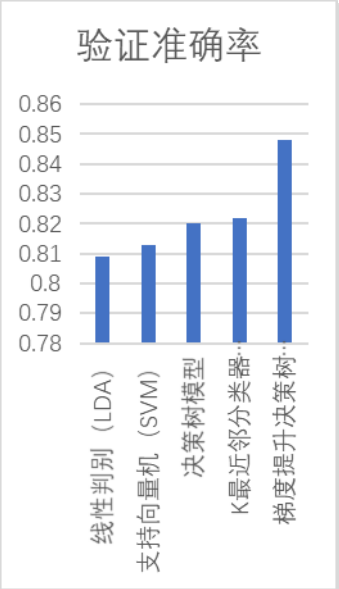

图 7.4 验证准确率

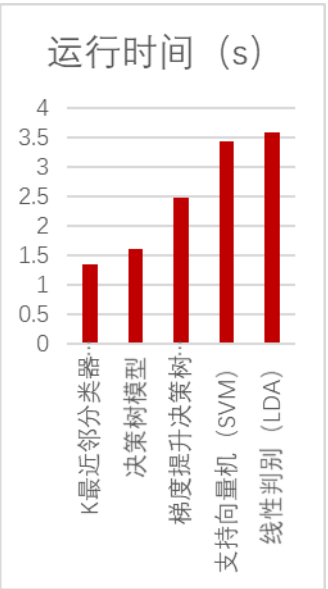

图 7.5 运行时间

八、问题四模型的建立与求解

8.1 问题实现

选用前 20、15、10 和 8 个变量分别进行基于遗传算法的 BP 网络训练，对训练好的模型使用 extremum 脚本输出最大值 Z_{max}，以及 X_1-X_i(其中第一次实验 i=1, 2，…，20；第二次实验 i=1, 2，…，15；第三次实验 i=1, 2，…，10；第四次实验 i=1, 2，…，8)。将生成的 X_1-X_i 输出到问题三训练的梯度提升决策树模型中计算 ADMET 离散值。四次实验的数据如下表 8.1 所示：

表 8.1 基于遗传算法的 BP 网络训练四次实验数据

实验次数 / 分子	第一次	第二次	第三次	第四次
LipoaffinityIndex	-3.732	-3.663	5.542	-4.030
MDEC-23	1.318	2.799e-07	6.197	16.198
SwHBa	-21.762	-18.615	-6.401	-21.762
MLogP	1.460	5.750	5.750	5.750
n6Ring	8.066e-06	3.531	3.169	0.756
nwHBa	4.174e-06	4.445e-07	7.596	8.712
nRing	7.000	7.000	7.000	5.793
C2SP2	10.864	6.702	19.929	3.384e-06
ETA_Psi_1	0.372	0.491	0.372	
nT6Ring	2.749	3.739	5.120	
BCUTp-1h	12.098	7.969		
minsBr	0.408	0.513		
naAromAtom	1.042e-05	1.139		
mindssC	0.434	-1.903		
WTPT-2	1.927	1.927		
nAromBond	7.145			
nBondsM	9.523			
minwHBa	-2.031			
ETA_Epsilon_3	0.453			
SsBr	0.681			
最大值 Z_{max}	6.582	6.205	15.656	6.230
ADMET 性质	01001	1 1000	11001	00000

8.2 结果分析

由上表可知，第二次实验，即选用变量为前 15 个变量时，此时的 ADMET 性质最贴合期望输出 11010。此时生物活性 pIC50 最大值为 6.205，各分子描述符的值见表 8.1。由于 ADMET 性质四代表该化合物的口服生物利用度而并非毒性，并且此决策已满足题干中至少三个性质较好的要求，所以第二列为实验的最优结果。

九、模型的分析与检验

9.1 误差分析

9.1.1 问题一的误差分析

问题一的误差，主要是由于数据携带信息的缺失造成的。虽然灰色理论可以用来处理信息缺失的情况，但并不能完全处理。

9.1.2 问题二的误差分析

问题二的误差主要来源于两方面。一方面，未对 Molecular_Descriptor_training 和 Molecular_Descriptor_test 里的数据进行清洗，剔除掉一些无效值。而是将全部数据都用于模型训练。从而造成预测值与期望值的拟合度偏低，影响后续继续预测的结果。另一方面，无法确定遗传算法预定的参数（如种群大小、进化代数）是否合适。尽管多次实验表明进化到 14 代以后适应度函数已经收敛，但是由于硬件设备的局限性，未能对更多的进化代数进行模拟。因而每次的实验结果都有很大的不确定性。只能从多次实验中选取最优的实验结果作为最后的实验数据输出。

9.1.3 问题三的误差分析

问题三误差存在于两个方面，一是数据集过小，模型无法学习到全部的信息，所以对于测试数据，会出错误的判断。二是，因为在算法的迭代过程中，会有一些信息的损失。在经过多轮的迭代过后进行累加。导致更大的信息流失，从而产生误差。

9.1.4 问题四的误差分析

问题四是问题二与问题三的综合结果，其误差产生原因与二者一致，这里不做过多赘述。

9.2 模型的检验

本文要解决的四个问题中灰色关联模型、基于遗传算法的神经网络模型、提升决策树模型在建立的过程中通过了相应软件的检验，具有一定的合理性。

9.3 模型的不足

在训练模型之前应定下数据筛选规则，对数据集进行清洗，然后训练网络，这样模型的拟合度会得到相应的提高；本文设定的仅为一个隐藏层的 BP 网络，可以设定不同结构的 BP 神经网络，分别考察模型预测的准确性；对初始权值阈值进行优化还可以使用粒子群算法，对比遗传算法优化的初始值，给予比较性评价并作出定量分析。

由于时间限制，并未对上述不足进行弥补。应在后续的时间里逐步进行模型完善。

十、模型的评价

10.1 模型优点

问题一指标筛选，构建了灰色关联模型，验证自变量与 pIC50 的相关度，通过排序确定生物活性影响的重要性因素指标。该方法克服了可线性相关以及贫信息的弊端，在非线性关系中具有较好的适用性；

问题二本研究运用遗传算法对模型进行求解，在算法设计时将构建的神经网络模型嵌入遗传算法中。通过变异机制避免算法陷入局部最优，搜索能力强，通过引入自然选择中的概率思想，使个体的选择具有随机性且可拓展性强，易于与其他算法进行结合使用；

问题三使用梯度提升决策树（GBDT）模型，可以灵活处理各种类型的数据，包括连续值和离散值。在相对少的调参时间情况下，预测的准确率也可以比较高。相比其他普通机器学习模型更有一些优越性。

10.2 模型缺点

问题一主要变量筛选，在筛选时是基于数据挖掘方式对自变量与 pIC50 间关系进行计算，属于定量分析。为考虑实际生产中各位变量对生物活性影响之间的关键关系，后续研究应采用定量与定性研究结合的方式进行关键变量筛选；

问题二算法内包含的交叉率、变异率、种群大小和进化代数的设定需要依靠经验确定试参确定，且参数对最终生成的最优染色体影响很大。此外，遗传算法也无法保证得到的解就一定是最优的，目前尚无证明超过多少代为最优解。它只能有限提高原有 BP 神经网络的预测精度，并不能把预测误差较大的 BP 神经网络优化为能够准确预测的 BP 神经网络；

问题三在模型使用时，需要对学习器产生的残差进行累加，因此调参数优化时，运行时间也会增加。同时学习器之间存在依赖关系，难以并行训练数据。

总体而言本文所建的模型和使用的算法可行性高，具有有效性，探索空间大，值得进一步的研究。

十一、参考文献

[1] 闫陆洲. 小麦抗旱相关农艺性状和生理生化性状的灰色关联度分析[J]. 农技服务, 2016, 33(006):99-99.

[2] 孙晓东, 焦玥, 胡劲松. 基于灰色关联度和理想解法的决策方法研究[J]. 中国管理科学, 2005, 013(004):63-68.

[3] 刘奕君, 赵强, 郝文利. 基于遗传算法优化 BP 神经网络的瓦斯浓度预测研究[J]. 矿业安全与环保, 2015(02):61-65.

[4] 李如平. BP 神经网络算法改进及应用研究[J]. 菏泽学院学报, 2016, 38(2):13-17.

[5] 潘昊, 王晓勇, 陈琼,等. 基于遗传算法的 BP 神经网络技术的应用[J]. 计算机应用, 2005, 25(012):2777-2779.

[6] 柯国霖. 梯度提升决策树(GBDT)并行学习算法研究. 2016.

[7] Fisher R A. The use of multiple measurements in taxonomic problems[J]. Annals of eugenics, 1936, 7(2): 179-188.

[8] Zhang M L, Zhou Z H. ML-KNN: A lazy learning approach to multi-label learning[J]. Pattern recognition, 2007, 40(7): 2038-2048.

[9] Breiman, Leo, Jerome H. Friedman, Richard A. Olshen and C. J. Stone. “Classification and Regression Trees.” (1983).

[10] Shawe-Taylor J, Cristianini N. An introduction to support vector machines and other kernel-based learning methods[M]. Volume, 2000.

附录 Matlab 源程序

问题一算法程序

```
% 灰度相关
N = xlsread("shuju1.xlsx",1)
colmin = min(N)
colmax = max(N)
N = rescale(N,'InputMin',colmin,'InputMax',colmax)
refer = N(:,1);
comparison = N(:,2:end);
comparison_sum = width(comparison);
for i = 1:comparison_sum
g(:,i)=comparison(:,i)-refer;
end
maxg = max(max(abs(g)));
ming = min(min(abs(g)));
p = 0.5
k=((ming+p*maxg)./(abs(g)+p*maxg));
ks1i_column=size(k,1);
r=sum(k)/ks1i_column;
[rs,rind]=sort(r,'descend');
```

问题二算法程序

```
%% BP 神经网络脚本
clear all
clc
load data1 input output
input_train=input(1:1800,:)';
input_test=input(1801:1974,:)';
output_train=output(1:1800,:)';
output_test=output(1801:1974,:)';
inputnum=20;
hiddennum=41;
outputnum=1;
[inputn,inputps]=mapminmax(input_train);
[outputn,outputps]=mapminmax(output_train);
net=newff(inputn,outputn,hiddennum,{'tansig','purelin'},'trainlm');
W1= net. iw{1, 1};
B1 = net.b{1};
W2 = net.lw{2,1};
B2 = net. b{2};
net.trainParam.epochs=1000;
net.trainParam.lr=0.01;
net.trainParam.goal=0.00001;
net=train(net,inputn,outputn);
inputn_test=mapminmax('apply',input_test,inputps);
an=sim(net,inputn_test);
test_simu=mapminmax('reverse',an,outputps);
error=test_simu-output_test;
%基于遗传算法的神经网络脚本
clc
clear
load data1 input output
```

```
inputnum=20;
hiddennum=41;
outputnum=1;
input_train=input(1:1800,:)';
input_test=input(1800:1974,:)';
output_train=output(1:1800)';
output_test=output(1800:1974)';
[inputn,inputps]=mapminmax(input_train);
[outputn,outputps]=mapminmax(output_train);
net=newff(inputn,outputn,hiddennum);
maxgen=15;
sizepop=40;
pcross=[0.5];
pmutation=[0.05];
numsum=inputnum*hiddennum+hiddennum+hiddennum*outputnum+outputnum;
lenchrom=ones(1,numsum);
bound=[-3*ones(numsum,1) 3*ones(numsum,1)];
individuals=struct('fitness',zeros(1,sizepop), 'chrom',[]);
avgfitness=[];
bestfitness=[];
bestchrom=[];
for i=1:sizepop

individuals.chrom(i,:)=Code(lenchrom,bound);
    x=individuals.chrom(i,:);

individuals.fitness(i)=fun(x,inputnum,hiddennum,outputnum,net,inputn,outputn);
end
[bestfitness bestindex]=min(individuals.fitness);
bestchrom=individuals.chrom(bestindex,:);
avgfitness=sum(individuals.fitness)/sizepop;
trace=[avgfitness bestfitness];
 for i=1:maxgen
     i

individuals=Select(individuals,sizepop);

avgfitness=sum(individuals.fitness)/sizepop;

individuals.chrom=Cross(pcross,lenchrom,individuals.chrom,sizepop,bound);

individuals.chrom=Mutation(pmutation,lenchrom,individuals.chrom,sizepop,i,maxgen,bound);
    for j=1:sizepop
        x=individuals.chrom(j,:);

individuals.fitness(j)=fun(x,inputnum,hiddennum,outputnum,net,inputn,outputn);
    end

[newbestfitness,newbestindex]=min(individuals.fitness);

[worestfitness,worestindex]=max(individuals.fitness);
    if bestfitness>newbestfitness
        bestfitness=newbestfitness;

bestchrom=individuals.chrom(newbestindex,:);
    end
    %individuals.chrom(worestindex,:)=bestchrom;
    %individuals.fitness(worestindex)=bestfitness;

avgfitness=sum(individuals.fitness)/sizepop;
    trace=[trace;avgfitness bestfitness];
end
x=bestchrom;
w1=x(1:inputnum*hiddennum);
B1=x(inputnum*hiddennum+1:inputnum*hiddennum+hiddennum);
w2=x(inputnum*hiddennum+hiddennum+1:inputnum*hiddennum+hiddennum+hiddennum*outputnum);
```

```
B2=x(inputnum*hiddennum+hiddennum
+hiddennum*outputnum+1:inputnum*hidden
num+hiddennum+hiddennum*outputnum+out
putnum);
net.iw{1,1}=reshape(w1,hiddennum,inp
utnum);
net.lw{2,1}=reshape(w2,outputnum,hidd
ennum);
net.b{1}=reshape(B1,hiddennum,1);
net.b{2}=B2;%
net.trainParam.epochs=100;
net.trainParam.lr=0.1;
%net.trainParam.goal=0.00001;
[net,per2]=train(net,inputn,outputn);
inputn_test=mapminmax('apply',input_te
st,inputps);
an=sim(net,inputn_test);
test_simu=mapminmax('reverse',an,outp
utps);
error=test_simu-output_test;
%求训练好的神经网络模型输出的极
大值
fun = @(z)-sim(net,z);
z0
=[-4.588879876,0.000,-21.762,1.460,0.000 ,0.
000 ,0.000 ,0.000 ,0.372 ,0.000 ,7.969 ,-0.513 ,
0.000 ,-1.903 ,1.927 ,0.000 ,0.000 ,-2.031 ,0.4
32 ,-0.513]';
A=[];b=[];Aeq=[];beq=[];
lb=[-4.588879876,0.000,-21.762,1.460,0.
000 ,0.000 ,0.000 ,0.000 ,0.372 ,0.000 ,7.969 ,
-0.513 ,0.000 ,-1.903 ,1.927 ,0.000 ,0.000 ,-2.
031 ,0.432 ,-0.513];
ub=[23.004 ,54.020 ,49.967 ,5.750 ,6.00
0 ,32.000 ,7.000 ,24.000 ,0.752 ,7.000 ,16.748
 ,0.408 ,30.000 ,2.026 ,2.142 ,34.000 ,34.000 ,
2.026 ,0.453 ,0.681 ];
[z,zmax]=fmincon( @(z)fun(z),z0,A,b,A
eq,beq,lb,ub);
zmax = -zmax;
% 预测对应的 50 个 pIC50 拟合值
load('input_yanzheng2')
predict=sim(net,input_yanzheng2)
result=mapminmax('reverse',predict,outp
utps);

%函数
%编码函数
function ret=Code(lenchrom,bound)
flag=0;
while flag==0
    pick=rand(1,length(lenchrom));

ret=bound(:,1)'+(bound(:,2)-bound(:,1))'.*pick
;
    flag=test(lenchrom,bound,ret);
end
%适应度函数
function error =
fun(x,inputnum,hiddennum,outputnum,net,inp
utn,outputn)
w1=x(1:inputnum*hiddennum);
B1=x(inputnum*hiddennum+1:inputnum
*hiddennum+hiddennum);
w2=x(inputnum*hiddennum+hiddennum
+1:inputnum*hiddennum+hiddennum+hidden
num*outputnum);
B2=x(inputnum*hiddennum+hiddennum
+hiddennum*outputnum+1:inputnum*hidden
num+hiddennum+hiddennum*outputnum+out
putnum);
net=newff(inputn,outputn,hiddennum);
net.trainParam.epochs=20;
net.trainParam.lr=0.1;
net.trainParam.goal=0.00001;
net.trainParam.show=100;
net.trainParam.showWindow=0;
net.iw{1,1}=reshape(w1,hiddennum,inp
utnum);
net.lw{2,1}=reshape(w2,outputnum,hidd
ennum);
net.b{1}=reshape(B1,hiddennum,1);
net.b{2}=B2;
net=train(net,inputn,outputn);
an=sim(net,inputn);
error=sum(abs(an-outputn));
%解码函数
function
```

```
ret=Decode(lenchrom,bound,code,opts)
    switch opts
        case 'binary' % binary coding
            for i=length(lenchrom):-1:1

data(i)=bitand(code,2^lenchrom(i)-1);

code=(code-data(i))/(2^lenchrom(i));
            end

ret=bound(:,1)'+data./(2.^lenchrom-1).*(boun
d(:,2)-bound(:,1))';
        case 'grey'     % grey coding
            for i=sum(lenchrom):-1:2

code=bitset(code,i-1,bitxor(bitget(code,i),bitg
et(code,i-1)));
            end
            for i=length(lenchrom):-1:1

data(i)=bitand(code,2^lenchrom(i)-1);

code=(code-data(i))/(2^lenchrom(i));
            end

ret=bound(:,1)'+data./(2.^lenchrom-1).*(boun
d(:,2)-bound(:,1))';

        case 'float'
            ret=code;
    end
    %选择函数
    function ret=select(individuals,sizepop)
    fitness1=10./individuals.fitness;
    sumfitness=sum(fitness1);
    sumf=fitness1./sumfitness;
    index=[];
    for i=1:sizepop
        pick=rand;
        while pick==0
            pick=rand;
        end
        for i=1:sizepop
            pick=pick-sumf(i);
            if pick<0
                index=[index i];
                break;
            end
        end
    end
    individuals.chrom=individuals.chrom(in
dex,:);
    individuals.fitness=individuals.fitness(in
dex);
    ret=individuals;
    %交叉函数
    function
ret=Cross(pcross,lenchrom,chrom,sizepop,bo
und)
     for i=1:sizepop
        pick=rand(1,2);
        while prod(pick)==0
            pick=rand(1,2);
        end
        index=ceil(pick.*sizepop);
        pick=rand;
        while pick==0
            pick=rand;
        end
        if pick>pcross
            continue;
        end
        flag=0;
        while flag==0
            pick=rand;
            while pick==0
                pick=rand;
            end

pos=ceil(pick.*sum(lenchrom));
            pick=rand;
            v1=chrom(index(1),pos);
            v2=chrom(index(2),pos);

chrom(index(1),pos)=pick*v2+(1-pick)*v1;

chrom(index(2),pos)=pick*v1+(1-pick)*v2;
```

```
flag1=test(lenchrom,bound,chrom(index(1),:))
;

flag2=test(lenchrom,bound,chrom(index(2),:))
;
                if     flag1*flag2==0
                       flag=0;
                else flag=1;
                end
            end
        end
    ret=chrom;
    %变异函数
    function
ret=Mutation(pmutation,lenchrom,chrom,size
pop,num,maxgen,bound)
    for i=1:sizepop
        pick=rand;
        while pick==0
            pick=rand;
        end
        index=ceil(pick*sizepop);
        pick=rand;
        if pick>pmutation
            continue;
        end
        flag=0;
            while flag==0
                pick=rand;
                while pick==0
                    pick=rand;
                end

pos=ceil(pick*sum(lenchrom));
                pick=rand;
                fg=(rand*(1-num/maxgen))^2;
                if pick>0.5

chrom(i,pos)=chrom(i,pos)+(bound(pos,2)-ch
rom(i,pos))*fg;
                else

chrom(i,pos)=chrom(i,pos)-(chrom(i,pos)-bou
nd(pos,1))*fg;
                end

flag=test(lenchrom,bound,chrom(i,:));
            end
    end
    ret=chrom;
    %测试函数
    function flag=test(lenchrom,bound,code)
    x=code;
    flag=1
```

问题三算法程序

```
    function [trainedClassifier,
validationAccuracy] =
trainClassifier1(trainingData, responseData)
    inputTable = array2table(trainingData,
'VariableNames', {'column_1', 'column_2',
'column_3', 'column_4', 'column_5',
'column_6', 'column_7', 'column_8',
'column_9', 'column_10', 'column_11',
'column_12', 'column_13', 'column_14',
'column_15', 'column_16', 'column_17',
'column_18', 'column_19', 'column_20'});
    predictorNames = {'column_1',
'column_2', 'column_3', 'column_4',
'column_5', 'column_6', 'column_7',
'column_8', 'column_9', 'column_10',
'column_11', 'column_12', 'column_13',
'column_14', 'column_15', 'column_16',
'column_17', 'column_18', 'column_19',
'column_20'};
    predictors = inputTable(:,
predictorNames);
    response = responseData;
    isCategoricalPredictor = [false, false,
false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false,
false, false];
```

```
% 训练分类器
% 以下代码指定所有分类器选项并训练分类器。
template = templateTree(...
'MaxNumSplits', 50);
classificationEnsemble = fitcensemble(...
predictors, ...
response, ...
'Method', 'AdaBoostM1', ...
'NumLearningCycles', 100, ...
'Learners', template, ...
'LearnRate', 0.1, ...
'ClassNames', [0; 1]);
% 使用预测函数创建结果结构体
predictorExtractionFcn = @(x) array2table(x, 'VariableNames', predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));
% 向结果结构体中添加字段
trainedClassifier.ClassificationEnsemble = classificationEnsemble;
trainedClassifier.About = '此结构体是从分类学习器 R2021b 导出的训练模型。';
trainedClassifier.HowToPredict = sprintf('要对新预测变量列矩阵 X 进行预测，请使用: \n yfit = c.predictFcn(X) \n 将 "c" 替换为作为此结构体的变量的名称，例如 "trainedModel"。\n \nX 必须包含正好 20 个列，因为此模型是使用 20 个预测变量进行训练的。\nX 必须仅包含与训练数据具有完全相同的顺序和格式的\n 预测变量列。不要包含响应列或未导入 App 的任何列。\n \n 有关详细信息，请参阅 <a href="matlab:helpview(fullfile(docroot, "stats", "stats.map"), "appclassification_exportmodeltoworkspace")">How to predict using an exported model</a>。');
% 提取预测变量和响应
% 以下代码将数据处理为合适的形状以训练模型。
%
% 将输入转换为表
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10', 'column_11', 'column_12', 'column_13', 'column_14', 'column_15', 'column_16', 'column_17', 'column_18', 'column_19', 'column_20'});
predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7', 'column_8', 'column_9', 'column_10', 'column_11', 'column_12', 'column_13', 'column_14', 'column_15', 'column_16', 'column_17', 'column_18', 'column_19', 'column_20'};
predictors = inputTable(:, predictorNames);
response = responseData;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false];
% 执行交叉验证
partitionedModel = crossval(trainedClassifier.ClassificationEnsemble, 'KFold', 5);
% 计算验证预测
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);
% 计算验证准确度
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');



clear;clc;
L = xlsread("问题三数据.xlsx",1);%训练集响应向量集
SR = xlsread("最优解数据.xlsx",1)%第四问最优解
```

```
H = xlsread("问题一结果.xlsx",1,'B2:U1975');

for i = 1:5
Q = L(:,i);
[trainedClassifier, validationAccuracy] = trainClassifier1(H, Q);
    yfit(:,i) = trainedClassifier.predictFcn(SR);
    end

    V = xlswrite("第三问结果.xls",yfit)
```