具有遗传性疾病和性状的遗传位点分析

田一丁，曹赟，蔡涵
上海交通大学 APEX 实验室
{killa, yuncao, hcai}@apex.sjtu.edu.cn

摘要

统计遗传学是一门随着计算机发展及机器学习技术发展而兴起的学科，主要研究领域是分析染色体、基因及位点等遗传信息载体对某些遗传性状或遗传疾病的影响。机器学习技术在此领域具有重要的作用，例如，利用机器学习技术，训练根据样本的基因信息，预测样本患病概率的模型，或者将基因信息视为特征，通过特征选择方法判断某些疾病的致病基因，达到控制疾病的目的。本文提出了一种利用统计学及机器学习方法，筛选对遗传性状和疾病有影响的位点和基因的方法，并对此方法进行了评估。

关键词：遗传位点，遗传疾病，统计遗传学，机器学习，特征选择

1. 背景介绍

我们已经知道人类的遗传性状由染色体上的 DNA 分子携带，DNA 分子是一种长链双螺旋结构，其包含许许多多的碱基对，这些碱基对由 A、T、C、G 四种碱基构成。人类在长期繁衍中会发生一些变异或者突变，造成性状的多样性。DNA 上有一些碱基对容易发生突变，这些碱基对被称为位点。

即使人类早在一百多年前就开始对遗传学进行探索和研究，但是至今我们对 DNA 分子，基因这些东西所知仍然甚少。现如今，随着计算机技术的不断发展，计算速度的不断上升以及统计学习的发展，生物遗传学进入了一个新的发展领域——统计遗传学。从大量的病例样本中学习哪些是引起某些疾病的致病基因是统计遗传学中一个重要的课题。

位点在 DNA 中占到大约 1/1000，出现的频率比较高，每个位点变异的方式也有多种，所以近年来被研究人员视为重要的研究对象。大量的研究也表明位点或者包含很多位点的基因与一些遗传病的有无十分相关。位点通常由两个碱基表示，例如 TT、CC 等，一个给定的位点最多可能有三种编码。

2. 题目介绍及分析

本题目是在遗传统计学的背景下衍生的题目，共有四个小题目，已知某种遗传病，给定一些患病样本和健康样本，以及这些样本某些位点的碱基编码，要求分析位点们与遗传病的关系。

第一题要求我们将位点的碱基编码转化为可计算的数值编码；第二题要求我们分析某个或某些位点与这种遗传病相关；第三题将所有位点进行划分，分为不同的一些基因，要求我们分析遗传病与基因的关系；第四题给出十种症状，要求我们分析这些症状与位点之间的关联。

题目中给出的数据包含 1000 个样本，其中包含 500 个健康样本和 500 个患病样本。每个样本给出其 9445 个位点的编码信息，以及十种性状信息。同时给出 300 个基因以及这些基因包含的位点。数据中的每个位点都有一个唯一的名称。

我们经过分析后发现，由于样本的患病与否可以用 0 或 1 来表示，这十分符合机器学习中二分类问题的输出，同时 1000 个样本可以作为训练数据和检验数据，故这个问题可以被看作一个机器学习问题，并且是一个监督学习问题（**Supervised Learning**）。每个样本的位点们可以看作这个样本的特征（**Feature**），分析哪些位点与疾病相关即分析哪些特征对最终结果的输出会产生影响，所以实际上这是一个特征选择（**Feature Selection**）问题。直观的选择思路便是选择那些使用后能够明显提升训练效果的特征。

3. 解题思路

发现了问题的本质之后我们就可以着手解决问题了。在这之前需要确定使用的编程语言以及工具，综合熟练程度、运行速度、现有工具丰富度等方面因素之后，我们决定使用 Python 作为我们的编程语言，并使用其 **scikit-learn** 工具包进行机器学习。

3.1 问题一：碱基编码

目前生物学中以及本题给出的数据中都是使用 A、T、G、C 四个字母对碱基进行编码，这种编码方式在碱基表示及描述中起到了很好的作用，但是文字编码并不能用于计算，所以我们需要一种利于计算的数值编码方式来对碱基进行表述。由于每个位点的碱基对有且只有三种可能的值，所以一种比较直观的编码方法是使用 1、2、3，三个数字来表示三种可能的值，但是这种编码方式存在一个致命的问题：这种编码下三种碱基对的地位并不是相同的，比如 2 所对应的碱基对在数值上是 1 所对应的碱基对的两倍，这是不科学的。另外这种编码方式导致三种可能的碱基对在它们的空间（一维空间）上两两之间的距离不相等，然而作为特征，这三种碱基对之间应该没有距离差。

在简单对比了几种编码方式之后，我们最终选择了使用三维 one-hot 向量表示碱基对。one-hot 向量是机器学习中一种经典的数据表示方式，向量中的每一维只有 0 和 1 两种取值，一个向量中只能有一维的值为 1，其他值均为零。在这里，向量中的一维代表碱基对的一个可能取值，如果向量中某一位为 1，表示碱基对的取值是这个 1 所对应的取值。比如某个位点的碱基对可能有三种取值：TT、TC、CC，则构建一个三维 one-hot 向量，如果碱基对的取值是 TT，则其对应的 one-hot 为 (1, 0, 0)；TC 对应 (0, 1, 0)，依次类推。这样三种碱基对应的值没有数值上的差别，并且在三维空间中它们两两之间的距离相等。

对于基因来说，一个基因包含多个位点，将这些位点的三维 one-hot 向量拼接起来得到整个基因的编码，如一个包含三个位点的基因可以表示为（0,0,1 ， 0,1,0 ， 0,0,1）这样一个 9 维向量，该编码下，在使用机器学习模型学习权值参数后，可以根据权值大小得到每一个位点的任何一个碱基对对疾病的影响程度。

3.2 问题二/三：分析遗传病与位点及基因的关系

对于问题二与问题三，由于其输出（是否患病）只有两种取值：1 或者 0，故我们将其视作一个分类问题。常用的监督学习的分类器有逻辑回归（**Logistic Regression**）、支持向量

机（**Support Vector Machine**）、决策树分类器（**Decision Tree Classifier**）、随机森林分类器（**Random Forest Classifier**）、梯度推进分类器（**Gradient Boosting Classifier**）等。

逻辑回归分类器[1]是一种比较简单的线性分类器，它为每一个输入训练一个权重，加权求和之后使用一个 **sigmoid** 函数进行分类，**sigmoid** 函数一般使用 $S(t)=\frac{1}{1+e^{-t}}$ 或者 tanh 函数。

支持向量机分类器[2]是一种多种维度适用的分类器模型，这得益于支持向量机的核函数思想，它的原理是将数据映射到一个高维空间中，并使用一个高维空间中的超平面来分类。在迭代过程中支持向量机不断寻找与两边数据点的间隔最大的超平面。

决策树[3]分类器是一种树模型分类器，其中树的每个决策节点由一些简单的决策规则构成，可以依靠计算条件概率的方法来构建。

随机森林[4]分类器是一个决策树的集合，其中每个决策树使用从所有特征中随机挑选的特征来对决策树进行训练，分类时使用决策森林分类结果中的众数作为最终的分类结果。

梯度推进分类器[5]实际上是一种框架，它会在多次迭代中训练许多效果较差的分类器，每次迭代都会给上次迭代结果较差的训练样本加上更高的权重，最后将这些分类器组合（加权求和或投票）后成为一个效果较好的分类器，在训练新的分类器时，以残差减少的梯度方向进行。

在实验过程中，我们分别尝试了这几种分类器模型并比较了它们的结果（见第四章），由于随机森林分类器训练的效率太低，而决策树分类器的效果欠佳，故最终放弃了这两种模型。

我们首先将所有位点都作为特征进行训练，在测试集上进行测试后发现准确度很低，因此如果在所有位点的基础上进行特征选择和排除，一来由于特征数量过于庞大，二来由于准确度很低，移除一些特征可能会导致结果变化并不明显，故考虑在初期先对特征进行筛选。我们针对每个特征单独训练了逻辑回归分类器，并对它们的测试结果进行了排序，由此选取单独条件下对结果影响最大的若干个位点作为初步筛选特征的结果。

在进一步的特征筛选中，我们使用了递归特征消除[6]（**Recursive Feature Elimination**）的方法，这是一种特征消除的框架，其思想为在迭代过程中不断剔除不重要的特征（具体表现为特征对应的权重较小）。具体方法是在每一轮迭代过后，排除对应权重最小的一个或几个特征，然后重新进行训练进入下一次迭代，直到特征数量达到要求的值为止。如果不确定最优的特征数量，可以结合交叉验证寻找最优的特征数量。递归特征消除这种方法是一种从模型顶层消除特征的方法，其弱点在于它的效果对模型的稳定性依赖较强，所在模型设计的过程中一定要进行正则化操作，防止过拟合。

使用递归特征消除的方法，我们成功筛选出了与遗传病最相关的位点和个基因。（见第四章）

3.3 问题四：分析不同性状与位点关系

人体的许多遗传疾病和性状是有关联的，如高血压，心脏病、脂肪肝和酒精依赖等。科研人员往往把相关的性状或疾病放在一起研究，这样能提高发现致病位点或基因的能力。我们首先对问题四中的十种疾病的数据进行了统计分析。结果如下表 1 及表 2。

	Label 1	Label 2	Label 3	Label 4	Label 5	Label 6	Label 7	Label 8	Label 9	Label 10
正例出现计数	500	500	500	500	500	500	500	500	500	500

表 1 各症状出现次数统计

	Label 1	Label 2	Label 3	Label 4	Label 5	Label 6	Label 7	Label 8	Label 9	Label 10
Label 1	0	429	435	427	421	420	435	428	436	432
Label 2	0	0	428	422	420	423	429	433	434	437
Label 3	0	0	0	425	431	423	431	432	428	437
Label 4	0	0	0	0	421	419	430	433	430	428
Label 5	0	0	0	0	0	437	420	427	417	423
Label 6	0	0	0	0	0	0	421	427	419	417
Label 7	0	0	0	0	0	0	0	434	430	440
Label 8	0	0	0	0	0	0	0	0	432	440
Label 9	0	0	0	0	0	0	0	0	0	436
Label 10	0	0	0	0	0	0	0	0	0	0

表 2 症状间同时出现次数统计

表 1 表明各症状出现的次数非常的平均。表 2 表明任何一对症状同时出现的概率非常高而且，各症状对的同时出现率也非常平均。

因此我们可以做出一个大胆的假设，将每一种症状认为是同一遗传病下的病理表现之一。如果把无任何症状作为 0， 有所有 10 个症状作为 1，那么很自然的我们可以将每一个症状的权重设置为 0.1 来建模该遗传病的严重程度。比如 001001100 症状的严重程度就是 0.3，这样我们把一个 10 个类的复杂分类问题简化为 1 个 0-1 数值标注的回归问题。这样我们就能利用机器学习中的各种经典回归模型进行建模，来拟合该数值标注数据。

我们分别尝试了线性回归（**Linear Regression**），岭回归（**Ridge Regression**），Lasso 回归（**Lasso Regression**），支持向量机（**Support Vector Regressor**），梯度推进回归（**Gradient Boosting Regression Tree**）等经典回归模型，具体结果见章节 4。

4. 实验结果及分析

4.1 问题二：染色体位点筛选

（1）**一次过滤**：首先我们对每一个位点各自训练一个分类模型，我们采用了 4 种分类器，包括：

A.逻辑回归模型带 l1 惩罚项（LR_l1）

B.逻辑回归模型带 l2 惩罚项（LR_l2）

C.支持向量机模型（LinearSVC）

D.梯度推进决策树（GB）

每种分类器对所有 9445 个位点训练一组参数，共学习了 9445*4 个分类模型。对每种分类模型，统计各位点的预测准确率分布如下图 1.

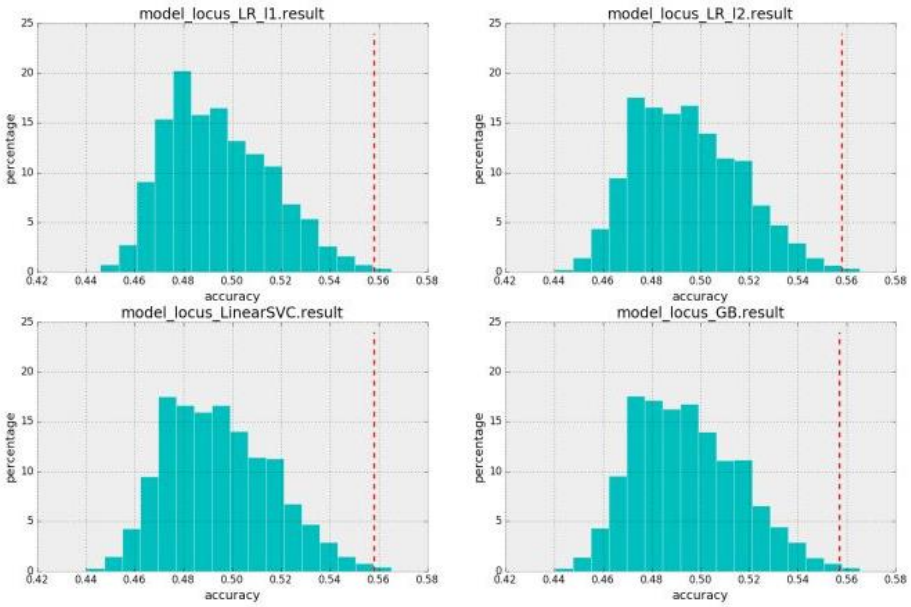

图 1.四种模型下位点准确度度量分布图

可以看到每一种模型的预测准确率分布都很相近，大量的位点的预测准确率在比较小的位置，只有少数能达到比较好的预测效果，因此我们将绝大部分的较差的位点进行一次过滤，只留下 30 个预测准确率最高的位点，图 1 中的红线代表了前 30 个位点的分界线，可以看到，只有最优秀的位点进入了进一步的筛选。

（2）二次过滤：

我们采用递归特征消除（**Recursive Feature Elimination**）方法来进行二次过滤，首先用所有一次过滤剩下的位点的特征向量组合成总特征向量。每一轮中，尝试删除每一个位点的特征，然后进行 4 种模型的训练，在各自模型下，去掉删除后使预测准确率提升最大的位点的特征，得到新的总特征向量。不断迭代直到删除剩下的任何一个向量都会使预测准确度下降，那么表明剩下的每一个位点都是对该疾病的预测至关重要的。

表格 3 是交叉验证下，最终预测准确率最高的模型：支持向量机模型 LinearSVC 每一轮递归特征消除的过程记录。（其他模型的 RFE 记录表见附录）

	Score 30	Score 29	Score 28	Score 27	Score 26	Score 25
Feature 0	0.726	0.708	0.683	0.701	0.697	0.699
Feature 1	0.726	0.693	0.689	0.698	0.709	0.704
Feature 2	0.726	0.726	0.727	0.728	0.727	0.729
Feature 3	0.726	0.708	0.713	0.72	0.715	0.712
Feature 4	0.726	0.71	0.709	0.716	0.715	0.719
Feature 5	0.726	**0.729**				
Feature 6	0.726	0.717	0.718	0.73	0.721	**0.73**
Feature 7	0.726	0.72	0.713	0.718	0.714	0.714
Feature 8	0.726	0.716	0.707	0.717	0.72	0.715
Feature 9	0.726	0.723	0.72	0.723	0.726	0.725
Feature 10	0.726	0.724	**0.73**			
Feature 11	0.726	0.718	0.714	0.721	0.721	0.715
Feature 12	0.726	0.723	0.718	0.719	0.732	0.728
Feature 13	0.726	0.719	0.724	0.729	0.723	0.725
Feature 14	0.726	0.713	0.71	0.72	0.719	0.719
Feature 15	0.726	0.708	0.701	0.709	0.713	0.709
Feature 16	0.726	0.714	0.717	0.716	0.727	0.724
Feature 17	0.726	0.71	0.718	0.718	0.722	0.72
Feature 18	0.726	0.715	0.726	0.722	0.732	0.723
Feature 19	0.726	0.717	0.71	0.714	0.716	0.713
Feature 20	0.726	0.728	0.727	0.727	**0.734**	
Feature 21	0.726	0.711	0.707	0.71	0.726	0.72
Feature 22	0.726	0.721	0.73	0.725	0.727	0.725
Feature 23	0.726	0.707	0.709	0.717	0.713	0.714
Feature 24	0.726	0.706	0.709	0.717	0.718	0.714
Feature 25	0.726	0.722	0.72	0.726	0.728	0.728
Feature 26	0.726	0.724	0.725	**0.733**		
Feature 27	0.726	0.716	0.71	0.714	0.722	0.716
Feature 28	0.726	0.724	0.723	0.722	0.731	0.73
Feature 29	0.726	0.722	0.727	0.726	0.723	0.721

表 3 Linear SVC 模型递归特征消除迭代准确率

从表 3 中可以看到，一次过滤后总特征模型的预测准确率是 72.6%，与每个位点独立模型的最大准确率 58.6%有非常大的提升，说明一次过滤的效果非常好。此外，二次过滤中每一次迭代都使预测准确率进一步提升，最终留下了 26 个位点，交叉验证下准确率达到 73.4%

（3）综合模型

我们选取交叉验证结果最好的模型作为我们的最终模型，并且二次过滤得到的位点作为最终筛选出来的影响该疾病的位点。

最终模型的预测值分布图如下图 2

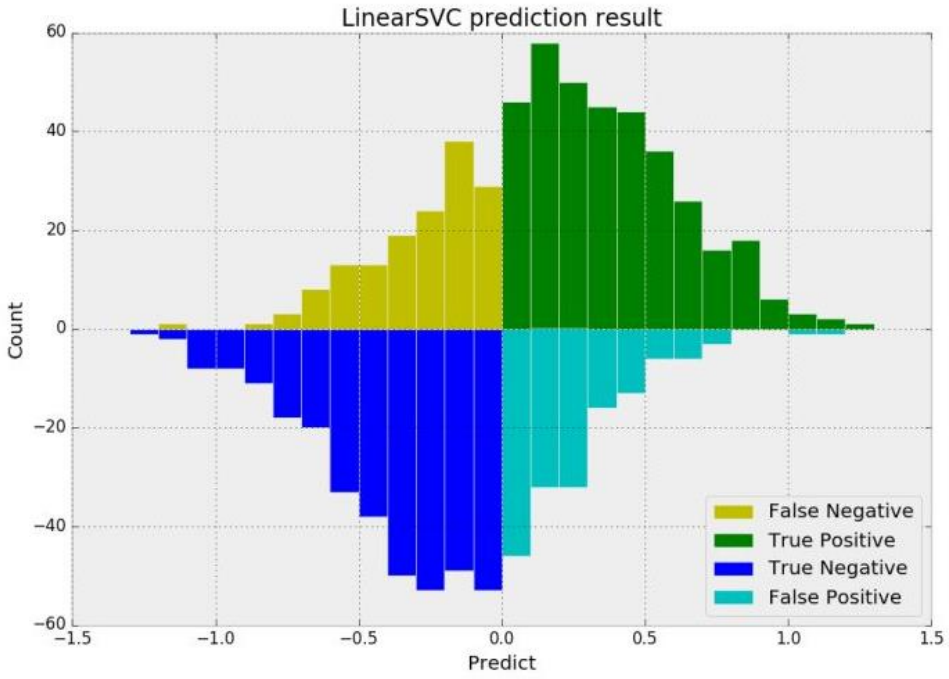

图 2：LinearSVC 模型二次过滤位点集下的预测结果分布

最终模型的交叉验证下的预测准确率（Accuracy）达到 73.4%，相较单个位点的最大值 58.6%有显著提升。最终选取的位点编号及对应名称为：

No	ID	位点名称	No	ID	位点名称
1	2937	rs2273298	14	7974	rs4585968
2	8379	rs7543405	15	2824	rs6660810
3	3122	rs4845952	16	5181	rs12097284
4	5224	rs2095518	17	4725	rs9442200
5	7113	rs586589	18	498	rs4648390
6	2793	rs9662275	19	5791	rs2473808
7	4931	rs2143810	20	6779	rs2744720
8	961	rs4391636	21	6793	rs2807345
9	829	rs12117836	22	8397	rs313990
10	2001	rs623974	23	871	rs4648347
11	3931	rs7543809	24	1274	rs364642
12	2192	rs845218	25	5809	rs16862684
13	5368	rs4310409	26	7461	rs1142057

表 4.最终选取的位点表

4.2 问题三：基因筛选

通过问题一的编码设计，问题三同样可以用问题二类似的解法，只不过这里的单位元变成了包含多个位点的基因。

（1）同样的，首先我们对 300 个基因建模，包括 4 种分类模型，共学习了 1200 个模型，并进行预测，准确率分布如下图 3：

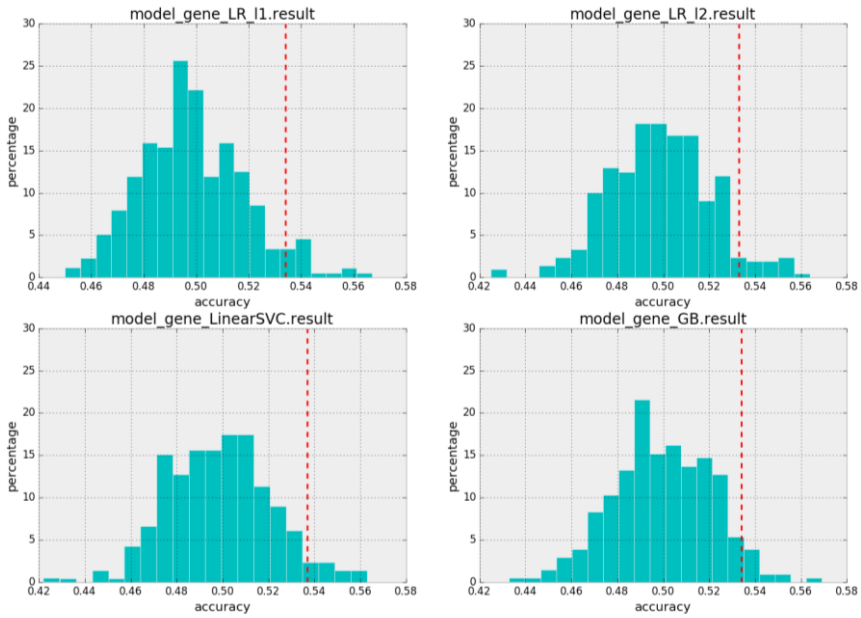

图 3.四种模型下基因准确率度量分布图

同样的大量的基因的模型准确率较差，少数的有较高的准确率，我们选取前 15 个基因作为一次过滤的结果，图中的红线代表前 15 个基因准确率的分界线。

（2）然后我们用递归特征消除 RFE 来进行二次筛选，过程表如下表 5：

	Score 15	Score 14	Score 13	Score 12	Score 11
Feature 0	0.618	0.614	0.6	0.598	0.602
Feature 1	0.618	0.628	0.622	0.637	**0.641**
Feature 2	0.618	0.618	0.623	0.621	0.631
Feature 3	0.618	0.618	0.626	0.64	0.635
Feature 4	0.618	0.612	0.623	0.63	0.629
Feature 5	0.618	0.625	**0.64**		
Feature 6	0.618	0.605	0.618	0.614	0.619
Feature 7	0.618	**0.632**			
Feature 8	0.618	0.629	0.627	0.618	0.622
Feature 9	0.618	0.627	0.629	0.635	0.636
Feature 10	0.618	0.627	0.623	0.628	0.625
Feature 11	0.618	0.608	0.612	0.622	0.606
Feature 12	0.618	0.62	0.614	0.614	0.62
Feature 13	0.618	0.626	0.633	**0.643**	
Feature 14	0.618	0.618	0.618	0.624	0.633

表 5. LR_l1 模型递归特征消除迭代准确率

（3）最终我们采用 Logistic Regression with L1 norm 模型，得到的最终预测准确率达到 64.3%，相比单个基因下的最高预测准确率 56.7%有显著提升。

最终选择了 12 个基因，基因编号为：

No	ID	No	ID
1	101	7	41
2	34	8	249
3	37	9	103
4	266	10	216
5	3	11	264
6	161	12	125

表 6.最终选取的基因编号

4.3 问题四：多性状建模

多性状建模采用对 10 个症状的标注进行加权的方式，把原多标注问题，变为单值标注的回归问题。

我们使用均方差(mean squared error)作为评价回归模型的方法。

如下图 4 是使用 Ridge Regression 模型对每个基因独立建模的均方差分布:

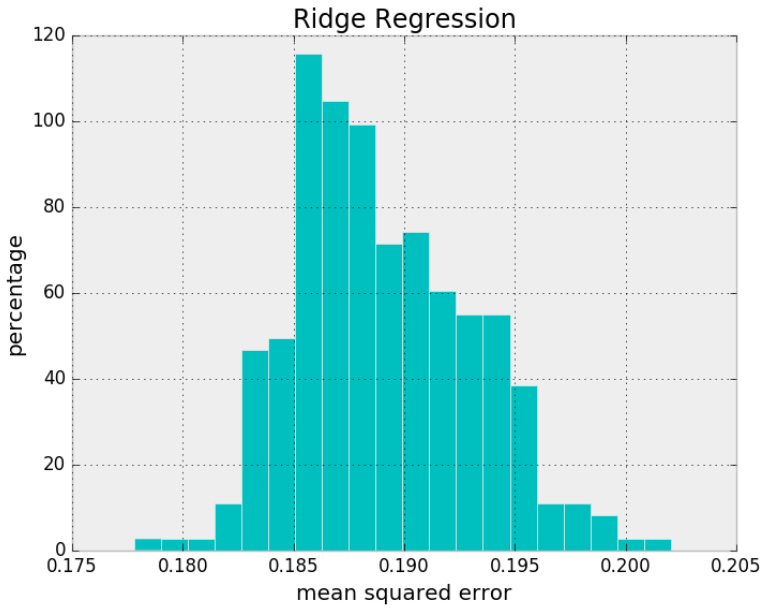

图 4.基因的均方差数据统计信息

同样选取前 15 个基因进入二次筛选。

	mse15	mse14	mse13	mse12	mse11
0	0.1674	0.1726	0.1707	0.1698	0.1679
1	0.1674	0.1699	0.1679	0.1675	0.1676
2	0.1674	0.1716	0.169	0.1675	0.1666
3	0.1674	0.1689	0.1668	0.1671	0.1664
4	0.1674	0.166	**0.1646**		
5	0.1674	0.1689	0.1669	0.1669	0.1666
6	0.1674	0.1686	0.1667	0.1663	0.1652
7	0.1674	0.1677	0.1655	0.1657	0.1648
8	0.1674	0.1674	0.1653	**0.1639**	
9	0.1674	0.1667	0.1648	0.1647	0.1646
10	0.1674	**0.1652**			
11	0.1674	0.1684	0.1665	0.1654	0.165
12	0.1674	0.1686	0.1672	0.1658	0.165
13	0.1674	0.1687	0.1654	0.1639	0.1642
14	0.1674	0.1664	0.1646	0.1644	**0.1641**

表 7. 基因多性状特征消除迭代结果

根据迭代结果，最终选取如下表中的基因：

No	ID	No	ID
1	165	7	11
2	88	8	65
3	214	9	0
4	238	10	94
5	187	11	213
6	55	12	15

表 8. 最终选取基因表

最终的均方差下降为 16.39。比单个基因的 17.8 下降显著。

5. 总结

本文分析了遗传性疾病和性状的遗传位点分析问题的本质，提出此问题可映射为机器学习问题中的特征选择问题。由此对问题建模，以位点（或者基因）作为机器学习问题中的特征，样本的性状或获病与否作为分类输出，尝试并比较了多种分类模型，利用机器学习的相关方法对遗传病及性状相关的位点和基因进行了筛选和分析，提出了一种分析人类 DNA 分子结构和表现性状对应关系的方法，并达到了较好的效果。

6. 鸣谢

感谢全国研究生数学建模竞赛组委会举办这次比赛，为我们提供一个检验自己知识水平和展示自己的平台，感谢所有工作人员的辛勤付出，感谢队友们的努力奋斗。另外需要特别感谢在机器学习领域中做出杰出贡献的科研工作者们，没有他们的铺垫我们不可能在科研的道路上走得如此稳定。

7. 参考文献

[1] Hosmer Jr D W, Lemeshow S. Applied logistic regression[M]. John Wiley & Sons, 2004.

[2] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. Neural processing letters, 1999, 9(3): 293-300.

[3] Quinlan J R. Induction of decision trees[J]. Machine learning, 1986, 1(1): 81-106.

[4] Ho T K. A data complexity analysis of comparative advantages of decision forest constructors[J]. Pattern Analysis & Applications, 2002, 5(2): 102-112.

[5] Wikipedia, Gradient boosting[DB/OL]. https://en.wikipedia.org/wiki/Gradient_boosting

[6] scikit-learn, Recursive Feature Elimination[EB/OL]. http://scikit-learn.org/stable/modules/feature_selection.html#recursive-feature-elimination

附录 A

第二/三题其他模型使用递归特征消除方法结果:

	Score 30	Score 29	Score 28	Score 27
Feature 0	0.705	0.703	0.702	0.689
Feature 1	0.705	0.701	0.699	0.698
Feature 2	0.705	0.72	0.715	0.716
Feature 3	0.705	0.703	0.705	0.706
Feature 4	0.705	0.709	0.719	0.726
Feature 5	0.705	0.722	0.72	0.712
Feature 6	0.705	0.709	0.722	0.724
Feature 7	0.705	0.715	0.712	0.704
Feature 8	0.705	0.713	0.699	0.715
Feature 9	0.705	0.711	0.711	0.725
Feature 10	0.705	0.711	**0.73**	
Feature 11	0.705	0.711	0.72	0.723
Feature 12	0.705	0.708	0.71	0.718
Feature 13	0.705	0.716	0.708	0.713
Feature 14	0.705	0.722	0.716	0.718
Feature 15	0.705	0.706	0.703	0.7
Feature 16	0.705	0.712	0.724	0.719
Feature 17	0.705	0.714	0.716	0.721
Feature 18	0.705	0.703	0.704	0.707
Feature 19	0.705	0.72	0.71	0.718
Feature 20	0.705	0.722	0.724	0.718
Feature 21	0.705	0.715	0.718	0.719
Feature 22	0.705	0.705	0.708	0.714
Feature 23	0.705	0.713	0.712	0.716
Feature 24	0.705	0.71	0.718	0.713
Feature 25	0.705	0.701	0.71	0.712
Feature 26	0.705	0.703	0.72	**0.729**
Feature 27	0.705	0.709	0.713	0.714
Feature 28	0.705	**0.723**		
Feature 29	0.705	0.704	0.715	0.713

表 A.1　位点分析 LR-l1 模型递归特征消除迭代准确率

	Score 30	Score 29	Score 28	Score 27	Score 26	Score 25
Feature 0	0.709	0.707	0.68	0.687	0.685	0.687
Feature 1	0.709	0.697	0.694	0.696	0.699	0.692
Feature 2	0.709	0.718	0.715	0.718	0.725	0.722
Feature 3	0.709	0.706	0.704	0.709	0.708	0.707
Feature 4	0.709	0.715	0.716	0.721	0.724	0.724
Feature 5	0.709	0.706	0.721	0.727	**0.733**	
Feature 6	0.709	0.708	0.717	0.713	0.717	0.721
Feature 7	0.709	0.713	0.709	0.721	0.731	0.71
Feature 8	0.709	0.713	0.703	0.706	0.709	0.71
Feature 9	0.709	0.716	0.723	0.714	0.716	0.718
Feature 10	0.709	0.706	0.714	0.712	0.719	0.724
Feature 11	0.709	0.715	0.723	0.72	0.722	0.726
Feature 12	0.709	0.702	0.716	0.725	0.726	0.718
Feature 13	0.709	0.713	0.707	0.704	0.713	0.713
Feature 14	0.709	0.7	0.721	0.728	0.728	**0.73**
Feature 15	0.709	0.711	0.708	0.711	0.716	0.709
Feature 16	0.709	0.695	0.701	0.706	0.71	0.697
Feature 17	0.709	**0.721**				
Feature 18	0.709	0.713	0.713	0.72	0.727	0.725
Feature 19	0.709	0.704	0.721	0.713	0.716	0.715
Feature 20	0.709	0.708	0.718	0.725	0.728	0.724
Feature 21	0.709	0.695	0.705	0.704	0.706	0.711
Feature 22	0.709	0.714	0.723	**0.729**		
Feature 23	0.709	0.707	0.702	0.711	0.712	0.71
Feature 24	0.709	0.711	**0.726**			
Feature 25	0.709	0.706	0.709	0.708	0.709	0.71
Feature 26	0.709	0.698	0.707	0.706	0.709	0.715
Feature 27	0.709	0.703	0.714	0.722	0.73	0.726
Feature 28	0.709	0.712	0.709	0.715	0.723	0.715
Feature 29	0.709	0.716	0.701	0.711	0.711	0.712

表 A.2　位点分析 LR-l2 模型递归特征消除迭代准确率

	Score 30	Score 29	Score 28	Score 27	Score 26
Feature 0	0.662	0.65	0.66	0.66	0.662
Feature 1	0.662	0.644	0.65	0.641	0.633
Feature 2	0.662	0.665	0.683	0.683	0.683
Feature 3	0.662	0.666	0.665	0.67	0.663
Feature 4	0.662	0.667	0.672	0.668	0.682
Feature 5	0.662	0.658	0.674	0.669	0.68
Feature 6	0.662	0.672	0.688	0.683	0.678
Feature 7	0.662	0.654	0.643	0.641	0.655
Feature 8	0.662	0.665	0.679	0.667	0.675
Feature 9	0.662	0.668	0.678	0.689	0.68
Feature 10	0.662	0.677	0.689	0.673	0.683
Feature 11	0.662	0.665	0.674	0.673	0.682
Feature 12	0.662	0.675	0.672	0.673	0.68
Feature 13	0.662	0.657	0.683	0.672	0.668
Feature 14	0.662	**0.684**			
Feature 15	0.662	0.658	0.683	0.682	0.684
Feature 16	0.662	0.67	0.684	0.68	0.66
Feature 17	0.662	0.66	0.672	0.674	0.682
Feature 18	0.662	0.663	0.681	0.685	0.68
Feature 19	0.662	0.675	0.677	0.677	0.673
Feature 20	0.662	0.669	0.671	0.677	**0.687**
Feature 21	0.662	0.677	0.671	0.674	0.681
Feature 22	0.662	0.652	0.658	0.657	0.643
Feature 23	0.662	0.664	0.68	**0.691**	
Feature 24	0.662	0.669	**0.689**		
Feature 25	0.662	0.654	0.671	0.681	0.676
Feature 26	0.662	0.653	0.684	0.669	0.685
Feature 27	0.662	0.673	0.674	0.669	0.666
Feature 28	0.662	0.66	0.68	0.686	0.664
Feature 29	0.662	0.673	0.685	0.677	0.675

表 A.3　位点分析 Gradiant Boosting 模型递归特征消除迭代准确率

附录 B

第四题其他模型使用递归特征消除方法结果:

	Score 14	Score 13	Score 12
Feature 0	0.533	0.524	0.542
Feature 1	0.533	0.53	0.535
Feature 2	0.533	0.55	0.543
Feature 3	0.533	0.54	0.543
Feature 4	0.533	0.526	0.541
Feature 5	0.533	0.536	0.545
Feature 6	0.533	0.558	0.548
Feature 7	0.533	0.543	0.523
Feature 8	0.533	0.541	0.53
Feature 9	0.533	0.561	0.567
Feature 10	0.533	0.533	0.526
Feature 11	0.533	0.53	0.533
Feature 12	0.533	0.544	**0.568**
Feature 13	0.533	**0.57**	
Feature 14	0.533	0.539	0.524
Max-value	0.533	0.57	0.568

表 B.1　基因分析 Gradiant Boosting 模型使用递归特征消除方法的结果

	Score 14	Score 13	Score 12	Score 11	Score 10
Feature 0	0.589	0.595	0.593	0.605	0.612
Feature 1	0.589	0.598	0.603	0.607	0.597
Feature 2	0.589	0.588	0.596	0.609	0.597
Feature 3	0.589	0.589	0.6	0.596	0.597
Feature 4	0.589	0.581	0.593	0.593	0.602
Feature 5	0.589	0.589	0.59	0.608	0.603
Feature 6	0.589	0.583	0.589	0.605	0.606
Feature 7	0.589	0.595	0.606	**0.615**	
Feature 8	0.589	0.597	0.596	0.591	0.601
Feature 9	0.589	0.593	**0.608**		
Feature 10	0.589	0.577	0.592	0.592	0.592
Feature 11	0.589	0.591	0.584	0.592	0.595
Feature 12	0.589	**0.607**			
Feature 13	0.589	0.595	0.587	0.594	**0.614**
Feature 14	0.589	0.585	0.577	0.579	0.594
Max-value	0.589	0.607	0.608	0.615	0.614

表 B.2　基因分析 LR-l2 模型使用递归特征消除方法的结果

	Score 14	Score 13	Score 12	Score 11	Score 10	Score 9
Feature 0	0.586	0.585	0.594	0.596	0.587	0.59
Feature 1	0.586	**0.601**				
Feature 2	0.586	0.581	0.6	**0.608**		
Feature 3	0.586	0.58	0.594	0.595	**0.615**	
Feature 4	0.586	0.578	0.592	0.591	0.59	**0.603**
Feature 5	0.586	0.575	0.597	0.593	0.59	0.601
Feature 6	0.586	0.576	0.595	0.589	0.587	0.593
Feature 7	0.586	0.574	0.583	0.564	0.584	0.585
Feature 8	0.586	0.581	**0.601**			
Feature 9	0.586	0.588	0.591	0.596	0.583	0.58
Feature 10	0.586	0.574	0.588	0.569	0.578	0.59
Feature 11	0.586	0.589	0.573	0.582	0.582	0.592
Feature 12	0.586	0.594	0.59	0.607	0.6	0.598
Feature 13	0.586	0.576	0.57	0.589	0.582	0.589
Feature 14	0.586	0.581	0.563	0.582	0.569	0.57
Max-value	0.586	0.601	0.601	0.608	0.615	0.603

表 B.3　基因分析 LinearSVC 模型使用递归特征消除方法的结果

附录 C

部分实验代码:

model_locus.py——首次过滤代码

```
import numpy as np
import cPickle
from load_data import DataLoader
from scipy.stats import pearsonr, spearmanr
from sklearn.cross_validation import cross_val_score, ShuffleSplit
#classification model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
#regression model
#from sklearn.linear_model import LinearRegression, Ridge, Lasso
#from sklearn.svm import SVR, LinearSVR
#from sklearn.tree import    DecisionTreeRegressor
#from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

data_loader = DataLoader()

genotype, genotable = data_loader.load_genotype()
phenotype = data_loader.load_phenotype()
feature_shape = len(genotype)
sample_size = len(phenotype)

#multi_phenos = data_loader.load_multi_phenos()
#multi_phenos_label = np.sum(multi_phenos, axis = 1) / 10.0

#locus importance
feature_score_acc = []
feature_score_f1 = []
C = 1
n_estimators = 100
print C, n_estimators
for feature_idx in range(feature_shape):
    if feature_idx%200==0:
        print feature_idx
    #get code ref
```

```
        sample_code = []
        for sample_idx in range(sample_size):
            sample_code.append(genotable[sample_idx][feature_idx])
        code_ref = set(sample_code)
        code2idx = {}
        for code in code_ref:
            code2idx[code] = len(code2idx)

        #feature
        sample_feature = np.zeros((sample_size, 3), dtype = np.int8)
        for sample_idx, code in enumerate(sample_code):
            sample_feature[sample_idx][code2idx[code]] = 1.

        #classification
        model = LogisticRegression(penalty = 'l1', C = C)
        #model = LinearSVC(C = C, random_state = 0)
        #model = SVC(C = C, kernel = 'rbf', degree = 3) #none linear
        #model = DecisionTreeClassifier(random_state = 0)
        #model = RandomForestClassifier(n_estimators = n_estimators, random_state =
0)
        #model = GradientBoostingClassifier(n_estimators = n_estimators,
random_state = 0)

        #regression
        #model = LinearSVR(C = C, random_state = 0)
        #model = SVR(C = C, kernel = 'rbf', degree = 3) #none linear
        #model = DecisionTreeRegressor(random_state = 0)
        #model = RandomForestRegressor(n_estimators = n_estimators, random_state
= 0)
        #model = GradientBoostingRegressor(n_estimators = n_estimators,
random_state = 0)

        cv = ShuffleSplit(len(phenotype), n_iter = 5, test_size = 0.2, random_state = 0)
        scores = []
        for train, test in cv:
            model.fit(sample_feature[train], phenotype[train])
            scores.append(model.score(sample_feature[test], phenotype[test]))

        #scores = cross_val_score(model, sample_feature, phenotype, scoring =
'accuracy' , cv = cv)
        feature_score_acc.append(np.mean(scores))
        #print sample_feature
        #print model.coef_
        #print scores
```

```
        #print 'avg acc:', np.mean(scores)
        #scores = cross_val_score(model, sample_feature, phenotype, scoring = 'f1' , cv
= cv)
        #feature_score_f1.append(np.mean(scores))
        #print scores
        #print 'avg f1:', np.mean(scores)
        #raw_input('pause')

print 'acc:'
print '    avg:', np.mean(feature_score_acc)
print '    max:', np.max(feature_score_acc)
#print 'f1:'
#print '    avg:', np.mean(feature_score_f1)
#print '    max:', np.max(feature_score_f1)

#save
with open('model_locus_LR_l1.result', 'w') as outfile:
    cPickle.dump((feature_score_acc, feature_score_f1), outfile)
```

RFE_classification.py——RFE 过滤代码

```
import numpy as np
import cPickle
from load_data import DataLoader
from matplotlib import pyplot as plt
from scipy.stats import pearsonr, spearmanr
from sklearn.cross_validation import cross_val_score, ShuffleSplit
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
#classification model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
#regression model
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR, LinearSVR
from sklearn.tree import    DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor

data_loader = DataLoader()

genotype, genotable = data_loader.load_genotype()
phenotype = data_loader.load_phenotype()
feature_shape = len(genotype)
sample_size = len(phenotype)

#multi_phenos = data_loader.load_multi_phenos()
#multi_phenos_label = np.sum(multi_phenos, axis = 1) / 10.0

result_file = 'filter_data/model_locus_LR_l1.result'
with open(result_file) as infile:
    feature_score_acc, feature_score_f1 = cPickle.load(infile)

#get top 30
start_size = 30
top_idx_list = []
for i in range(start_size):
    top_idx = np.argmax(feature_score_acc)
    #print 'top',i+1, ':', feature_score[top_idx]
    feature_score_acc[top_idx] = 0
    top_idx_list.append(top_idx)
```

```
#RFE
#make feature
cv = ShuffleSplit(len(phenotype), n_iter = 5, test_size = 0.2, random_state = 0)

C = 0.1
n_estimators = 40
print C, n_estimators

#classification
#model = LogisticRegression(penalty = 'l2', C = C)
model = LinearSVC(C = C, random_state = 0)
#model = SVC(C = C, kernel = 'rbf', degree = 3) #none linear
#model = DecisionTreeClassifier(random_state = 0)
#model = RandomForestClassifier(n_estimators = n_estimators, random_state = 0)
#model = GradientBoostingClassifier(n_estimators = n_estimators, random_state =
0)

def try_model(model, top_idx_list):
    combined_feature = np.zeros((sample_size, 3*len(top_idx_list)), dtype = np.bool)
    for idx, locus_idx in enumerate(top_idx_list):
        #get code ref
        sample_code = []
        for sample_idx in range(sample_size):
            sample_code.append(genotable[sample_idx][locus_idx])
        code_ref = set(sample_code)
        code2idx = {}
        for code in code_ref:
            code2idx[code] = len(code2idx)
        #feature
        for sample_idx, code in enumerate(sample_code):
            combined_feature[sample_idx][idx*3 + code2idx[code]] = 1.

    scores = []
    for train, test in cv:
        model.fit(combined_feature[train], phenotype[train])
        scores.append(model.score(combined_feature[test], phenotype[test]))
    #print len(top_idx_list), 'score:', np.mean(scores)
    return np.mean(scores)

def test_model(model, top_idx_list):
    print 'feat size:',len(top_idx_list)
    combined_feature = np.zeros((sample_size, 3*len(top_idx_list)), dtype = np.bool)
    for idx, locus_idx in enumerate(top_idx_list):
        #get code ref
```

```
            sample_code = []
            for sample_idx in range(sample_size):
                sample_code.append(genotable[sample_idx][locus_idx])
            code_ref = set(sample_code)
            code2idx = {}
            for code in code_ref:
                code2idx[code] = len(code2idx)
            #feature
            for sample_idx, code in enumerate(sample_code):
                combined_feature[sample_idx][idx*3 + code2idx[code]] = 1.

        model.fit(combined_feature, phenotype)
        pred_label = model.predict(combined_feature)
        pred_val = model.decision_function(combined_feature)
        print 'acc:', model.score(combined_feature, phenotype)
        print 'precision','recall','f1','accuracy'
        print precision_score(phenotype, pred_label), recall_score(phenotype,
pred_label), f1_score(phenotype, pred_label), accuracy_score(phenotype, pred_label)

        pos_eval = []
        neg_eval = []
        pos_bins = [0]*30
        neg_bins = [0]*30
        for i in range(sample_size):
            if phenotype[i]>0.5:
                pos_eval.append(pred_val[i])
                pos_bins[int(np.floor(pred_val[i]*10-15))]+=1
            else:
                neg_eval.append(pred_val[i])
                neg_bins[int(np.floor(pred_val[i]*10-15))]-=1
        plt.style.use('bmh')
        FN = plt.bar((np.arange(15)/10.-1.5), pos_bins[:15], 0.1, color = 'y')
        TP = plt.bar((np.arange(15)/10.), pos_bins[15:], 0.1, color = 'g')
        TN = plt.bar((np.arange(15)/10.-1.5), neg_bins[:15], 0.1, color = 'b')
        FP = plt.bar((np.arange(15)/10.), neg_bins[15:], 0.1, color = 'c')
        plt.ylabel('Count')
        plt.xlabel('Predict')
        plt.legend((FN[0],TP[0],TN[0],FP[0]),('False Negative', 'True Positive', 'True
Negative', 'False Positive'), loc = 4)
        plt.title('LinearSVC prediction result')
        plt.show()

max_score = try_model(model, top_idx_list)
print start_size, 'score:',max_score
```

```
while(len(top_idx_list)>1):
    feature_score = []
    for kick_idx in range(len(top_idx_list)):

        tmp_idx_list = top_idx_list[:kick_idx]+top_idx_list[kick_idx+1:]
        score = try_model(model, tmp_idx_list)
        print 'try kick\t%d\t%.4f'%(kick_idx,score)
        feature_score.append(score)
    best_kick = np.argmax(feature_score)
    if feature_score[best_kick]>=max_score:
        max_score = feature_score[best_kick]
        top_idx_list = top_idx_list[:best_kick]+top_idx_list[best_kick+1:]
        print 'Kick',best_kick
    else:
        break

print len(top_idx_list), 'final:', top_idx_list
print 'max acc:', max_score
```