参赛密码 

（由组委会填写）

**第十三届“华为杯”**

**全国研究生数学建模竞赛**

|  |  |
| --- | --- |
| **学 校** | **湖南师范大学** |
| **参赛队号** | **10542011** |
| **队员姓名** | **1.林益贤** |
| **2.陈绍鹏** |
| **3.刘婷** |

参赛密码 

（由组委会填写）

**第十三届“华为杯”全国研究生数学建模竞赛**

**具有遗传性疾病和性状的遗传位点分析**

**摘 要**

人类复杂疾病往往是由于多基因突变引发的，而这些发生突变的基因则通过相互作用影响疾病发展、调控生理活动。大量研究表明，人体的许多表型性状差异以及药物和疾病的易感性等都可能与某些位点相关联，或和包含有多个位点的基因相关联。因此，定位与性状或疾病相关联的位点在染色体或基因中的位置，能帮助研究人员了解性状和一些疾病的遗传机理，也能使人们对致病位点加以干预，防止一些遗传病的发生。

问题一针对不同的性状位点的碱基对类别的差异，采用一种新方法对碱基对进行编码，这种0,1,2编码方式不仅可以减少不同碱基对冗余信息的干扰，便于后续数据的处理。

问题二每个个体对应一个样本，每个个体的分类属性为是否患病，样本的每个特征代表一个SNP位点，其取值表示对个体SNP测试的结果。运用统计学以及机器学习的相关知识，对此数据进行分析，可以找到一个或多个SNP位点的子集，子集内的SNP位点或某些SNP位点的组合与所研究疾病有着统计意义上的强关联关系。针对具有遗传性疾病和性状的遗传位点进行相关性分析，通过降维处理筛选出111个的位点信息，然后ReliefF算法各自的特点及使用条件，提出改进的ReliefF和PCA算法结合起来进行SNP关联分析的算法。本文基于改进的ReliefF算法和PCA算法的结合来分析SNP关联分析，得出位点rs2273298、 rs7368252、rs2807345、rs1009806最有可能与疾病相关。

问题三对于问题二的求解，获得了每个位点对于疾病影响程度的权值。由于一个基因有多个位点，分析将一个基因理解为若干个位点组成的集合。将一个基因中的所有位点的权值累加起来，得到该基因的权值，该权值即表示该基因对疾病的影响程度。这样我们就得到了遗传疾病与该基因的关联性。选取权值大于设定的阈值（本文阈值设置为2）的基因，就找出了与疾病最有可能相关的19个基因（gene\_55、gene\_ 217、gene\_ 293等）。

问题四中找出跟性状有关联的位点，文中给出的是10个性状，然后每个性状有1000样本，考虑到10个性状和位点之间的关系，可以转化先考虑单个性状和位点之间的关系，而且性状的属性只有0和1两种，这样就类似于疾病和位点之间的关系，从而可以用问题二的ReliefF算法找出每个性状的关联位点。最后用统计分析的方法确定与10个性状都存在相关联的位点为：rs10909903,rs6686587,rs311469,rs3128326,rs608203,rs10799145,  
rs10492987,rs7545115,rs2501401,rs4314833,rs351619,rs770706 ,  
rs6678618,rs11247946 。

关键词：致病基因，PCA，ReliefF，全基因组关联性分析, SNP

**目 录**

[一、 问题的重述 2](#_Toc462133855)

[1.1 问题的背景 2](#_Toc462133856)

[1.2 问题的要求 2](#_Toc462133857)

[1.3 问题的分析 3](#_Toc462133858)

[1.3.1 对问题1的分析 3](#_Toc462133859)

[1.3.2 对问题2的分析 3](#_Toc462133860)

[1.3.3 对问题3的分析 4](#_Toc462133861)

[1.3.4 对问题4的分析 4](#_Toc462133862)

[二、 基本假设 5](#_Toc462133863)

[三、 符号说明和名词解释 5](#_Toc462133864)

[3.1 符号说明 5](#_Toc462133865)

[3.2 名词解释 5](#_Toc462133866)

[四、 模型的建立与求解 6](#_Toc462133867)

[4.1 问题1的分析与求解 6](#_Toc462133868)

[4.1.1 问题的分析与求解 6](#_Toc462133869)

[4.1.2 数据的采集与处理 6](#_Toc462133870)

[4.1.3 结论 7](#_Toc462133871)

[4.2 问题2的分析及求解 7](#_Toc462133872)

[4.2.1 问题的分析 7](#_Toc462133873)

[4.2.2 数据的采集与处理 8](#_Toc462133874)

[4.2.3 模型的建立与求解 8](#_Toc462133875)

[4.2.4 结论 14](#_Toc462133876)

[4.2.5 模型的进一步验证 15](#_Toc462133877)

[4.3 问题3的分析求解 18](#_Toc462133878)

[4.3.1 问题的分析 18](#_Toc462133879)

[4.3.2 数据的采集与处理 18](#_Toc462133880)

[4.3.3 模型的建立与求解 18](#_Toc462133881)

[4.3.4 结论 20](#_Toc462133882)

[4.3.5 模型的进一步验证 20](#_Toc462133883)

[4.4 问题4的分析与求解 21](#_Toc462133884)

[4.4.1 问题的分析与求解 21](#_Toc462133885)

[4.4.2 数据的采集与处理 21](#_Toc462133886)

[4.4.3 结论 22](#_Toc462133887)

[五、 模型评价与推广 23](#_Toc462133888)

[六、 参考文献 23](#_Toc462133889)

[七、 附录 24](#_Toc462133890)

1. 问题的重述
   1. 问题的背景

随着信息技术的发展，现代生物学越来越多地将这些技术用于大规模生物数据的收集、分析、挖掘等过程。大量计算机技术，特别是机器学习方法，被用来进行复杂疾病的分析。常见复杂疾病通常是在遗传因素和环境因素的共同作用下产生的，往往是受到多个基因的联合作用。

单核酸多态（SNP），是一种主要的遗传差异，已经成功被用来检测单基因疾病。然而大多数疾病是复杂疾病，这些疾病由多个基因以及基因与环境的交互作用导致。目前，已经有提出了不少研究SNP相互作用的算法。但是这些算法对于寻找3个及3个以上的有交互作用的SNP的组合并不是特别有效。本文借鉴了一种算法，可以同时分析所有的SNP。该方法将每个个体的SNP序列对应到高维空间中的一个点，这些点都携带相同大小的能量。然后寻找一个新的坐标系，使得疾病人群与健康人群的能量分布差异达到最大。根据这个新坐标系的信息，可以找到有交互作用的多位点的SNP的组合。在模拟数据上的实验显示本方法是有效的。

生物信息学是一门利用计算机科学、信息学、统计学和应用数学的方法研究生物问题的学科。现有的这一学科的研究基本上只是利用了信息技术与分子生物学。生物学技术能够生产大量的数据，其中包含很多嘈杂数据。生物学信息学就是利用计算工具从这些数据中提取相关的生物学习信息。GWAS是指在人类全基因组范围内找到与所有的序列变异，即单核苷酸多态性(SNP),从中筛选出于疾病相关的SNPS，利用大量的病例/对照组样本进行全基因遗传标记分型，找出所有变异的等位基因频率，用统计分析来找到以前未发现的基因，从而找与复杂疾病相关的遗传因素，得到一些复杂疾病致病机理的新的线索。

* 1. 问题的要求

问题一：用适当的方法，把genotype.dat中每位点的碱基(A、T、C、G)编码方式转化为数值编码方式，便于进行数据分析。

问题二：根据附录中1000个样本在某条有可能致病的染色体片段上的9445个位点的编码信息(见phenotype.txt文件)。设计或采用一个方法，找出某种疾病最有可能的一个或几个致病位点，并给出相关的理论依据。

问题三：同上题中的样本患有遗传疾病A的信息（phenotype.txt文件）。现有300个基因，每个基因所包含的位点名称见文件夹gene\_info中的300个dat文件，每个dat文件列出了对应基因所包含的位点(位点信息见文件genotype.dat)。由于可以把基因理解为若干个位点组成的集合，遗传疾病与基因的关联性可以由基因中包含的位点的全集或其子集合表现出来请找出与疾病最有可能相关的一个或几个基因，并说明理由。

问题四：在问题二中，已知9445个位点，其编码信息见genotype.dat文件。在实际的研究中，科研人员往往把相关的性状或疾病看成一个整体，然后来探寻与它们相关的位点或基因。试根据multi\_phenos.txt文件给出的1000个样本的10个相关联性状的信息及其9445个位点的编码信息(见genotype.dat)，找出与multi\_phenos.txt中10个性状有关联的位点。

* 1. 问题的分析
     1. 对问题1的分析

采用碱基(A、T、C、G)的编码方式来获取每个位点信息，因为染色体具有双螺旋结构，所以用两个碱基的组合表示一个位点的信息。对每个SNP位点进行编码，编码成最小等位基因的个数：如果最小等位基因为a,那么基因型AA,Aa,aa一个由(1,0,0),(0,1,0),(0,0,1)组成的向量。

* + 1. 对问题2的分析

为1000个样本在某条有可能致病的染色体片段上的9445个位点的编码信息，找出某种疾病最有可能的一个或几个致病位点，本文使用各种降维算法，数据挖掘算法与机器学习等算法寻找高阶SNP模式，这些模式在患病人群与健康人群中出现的频率应该存在非常大的差异，那么这些SNP模式就很有可能是致病的遗传标记。

SNP关联研究

无病跟患病个体

病例对照研究/随机人群队列

质量性状

数量性状

卡方检验

归因分析

方差分析

协方差分析

* + 1. 对问题3的分析

由于问题二的求解，获得了每个位点对于疾病影响程度的权值。由于一个基因有多个位点，所以我们有如下分析，将一个基因理解为若干个位点组成的集合。将一个基因中的所有位点的权值累加起来，得到该基因的权值，该权值即表示该基因对疾病的影响程度。这样我们就得到了遗传疾病与该基因的关联性。选取权值大的一些基因，就找出了与疾病最有可能相关的19个基因。

* + 1. 对问题4的分析

问题四中讨论10个相关性状的信息与其相关联的位点，给出的是10个性状，然后每个性状有1000样本，考虑到10个性状和位点之间的关系，可以转化先考虑单个性状和位点之间的关系，也就是转化为了当初的疾病和位点之间的关系，这里的不同之处在于当初疾病前500个是0，后500个是1，这里性状1000个样本的0和1是随机出现的，没有规律，先单个用算法计算出与每个性状相关的位点集合，最后统计存在这10个位点集合中共同位点作为整体相关联的性状位点。

1. 基本假设

(1).假设题目所给的数据真实可靠。

(2).假设间隔定义为一个样本的最近邻相同类别样本与最近邻不同类别样本的距离，对于K近邻分类器，假设间隔小于样本间隔。

1. 符号说明和名词解释
   1. 符号说明

|  |  |
| --- | --- |
| 符号 | 符号说明 |
| W[i] | 位点i的特征值权重值； |
| K | 为最邻近的数目； |
| M | 重复次数； |
| R | 随机选择的实例样本数； |
|  | P个随机变量； |
|  | 表示n个样本的位点为p个； |
| N | 训练样本N个特征； |

* 1. 名词解释

（1）MAF：等位基因最小频率；

（2）SNP：位点；

（3）ReliefF：机器学习中监督学习的算法；

1. 模型的建立与求解
   1. 问题1的分析与求解
      1. 问题的分析与求解

人的染色体都是成对出现的。如果得到每条染色体上对应SNP的信息，即单体型信息，可以推断出很多信息。但是现有的微矩阵技术单独得到每条染色体上对应SNP的信息很困难，而得到一对染色体上的SNP位点的混合信息相对容易，即基因型信息。目前各种研究机构已经积累了很多基因型数据，如何从这些数据中挖掘到有用的信息是非常重要的研究课题。本文中我们研究的重点也是基因以及位点型数据。当前我们主要研究二对等位基因的SNP，即给定一个SNP位点，它的值只有两种可能，主等位基因与次等位基因。*si*处的SNP的值g*i*有3种可能。g*i*=0，代表*si*处的两条染色体上都是主等位基因。*gi*=2，代表*si*处的两条染色体上都是次等位基因。*gi*=1代表*si*处的两条染色体上是一个主等位基因和一个次等位基因。如果一个人的DNA序列有*n*个SNP位点，那么其SNP数据可以表示为一个由0，1，2组成的向量。在我们的算法中在此基础上进行了一个变换，得到新的向量。 其变换如公式所示：



这种编码方式的是每个人的SNP序列对应一个点，这些点位于高维空间中的不同位置，它们携带相同的能量。如果使用0,1,2的编码方式，虽然它们表示的点的位置不一样，但是同时它们携带的能量也不一样，因而不符合我们算法的要求。我们使用hi表示第i个健康人的SNP序列，dj表示第j个患者的SNP序列。则健康人群的SNP数据可以表示为3n×t的矩阵H=(h1,h2,……,ht)，患病人群的SNP表示3n×r矩阵D=（d1,d2,……,dr）,其中t和r分别为两种人群人数。

* + 1. 数据的采集与处理

采用题目中给定的genotype.dat与phenotype.txt文件数据，genotype.dat文件中一个位点为碱基对构成，碱基对有3中可能，如AA、AT、TT，SNP位点数据可以表示为由0,1,2组成的向量。

MATLAB核心代码实现如下：

for j=1:1:9445

letter(1)='A';letter(2)='T';letter(3)='C';letter(4)='G';

judge(:,1)=strcmp(genotype\_cell(:,j),[letter(1),letter(1)]);

sum1(1)=sum(judge(:,1));

judge(:,2)=strcmp(genotype\_cell(:,j),[letter(2),letter(2)]);

sum1(2)=sum(judge(:,2));

judge(:,3)=strcmp(genotype\_cell(:,j),[letter(3),letter(3)]);

sum1(3)=sum(judge(:,3));

judge(:,4)=strcmp(genotype\_cell(:,j),[letter(4),letter(4)]);

sum1(4)=sum(judge(:,4));

[sum1\_sorted,sum1\_idx]=sort(sum1,'descend');

goodLetter(1)=letter(sum1\_idx(1));

goodLetter(2)=letter(sum1\_idx(2));

judge5=strcmp(genotype\_cell(:,j),[goodLetter(1),goodLetter(2)]);

sum2=sum(judge5);

judge6=strcmp(genotype\_cell(:,j),[goodLetter(2),goodLetter(1)]);

sum3=sum(judge6);

bp{1}=[goodLetter(1),goodLetter(1)];

bp{3}=[goodLetter(2),goodLetter(2)];

if sum2>=sum3

bp{2}=[goodLetter(1),goodLetter(2)];

else

bp{2}=[goodLetter(2),goodLetter(1)];

end

genotype\_cell(:,j)=strrep(genotype\_cell(:,j),bp(2),'100');

genotype\_cell(:,j)=strrep(genotype\_cell(:,j),bp(1),'010');

genotype\_cell(:,j)=strrep(genotype\_cell(:,j),bp(3),'001');

end

* + 1. 结论

SNP位点由碱基对构成，对碱基对进行编码，本文采用0,1,2的编码方式，以便后面数据的分析处理。

* 1. 问题2的分析及求解
     1. 问题的分析

鉴于复杂疾病通常是由基因网络上多个位点的变异引起，每个SNP位点的变异可能单独对疾病产生致病作用，也有可能每个位点的变异只有比较微弱的致病效应，但是通过和其他SNP位点的统计交互作用就能对疾病产生很强的致病作用。每个个体对应一个样本，每个个体的分类属性为是否患病，样本的每个特征代表一个SNP位点，其取值表示对个体SNP测试的结果。运用统计学以及机器学习的相关知识，对此数据进行分析，可以找到一个或多个SNP位点的子集，子集内的SNP位点或某些SNP位点的组合与所研究疾病有着统计意义上的强关联关系。本文提出的方法同时分析单个位点和多个位点的交互作用，识别对疾病有致病作用的SNP位点。

本文全基因组致病SNP位点识别主要包括下面几项工作：

降维（筛选出111个位点）；

改进的ReliefF算法（前10个权重相对大的特征值）；

PCA算法筛选出对疾病最有可能的4个位点SNP；

卡方检验；

（1） 先进行降维处理，对9445个位点进行筛选；

（2） 深入分析各种识别致病SNP位点方法的性能；

本文通过模拟数据集，对各种识别致病位点的方法进行比较，分析了卡方检验、ReliefF和PCA等方法在各种模型下识别致病位点的能力，分析这些方法在哪些模型下表现突出，或者在哪些模型下表现不足，从而为本文提出的方法提供依据。

（3） 提出结合ReliefF和PCA的识别方法；

本文提出了两种结合ReliefF和PCA的识别方法，这两种方法都是基于后向缩减的迭代策略，都能最终产生对所有SNP位点的特征值排名，可以利用这个排名去筛选得到致病位点。

（4） 验证方法的有效性；

问了验证提出方法的有效性，同时在大量模拟数据集和真实数据集进行验证，通过设置不同模拟数据集的模型，分析提出的方法在各种模型下的表现。

* + 1. 数据的采集与处理

采用题目中给定的genotype.dat与phenotype.txt文件数据，从phenotype.txt文件中1000个样本中前500样本是无病的，后500是患病的，分析此两文件数据的关联关系，从而分析出疾病最有可能是哪些位点造成的。

* + 1. 模型的建立与求解

本文先用最小等位基因频率MAF对SNP位点进行筛选，把筛选过后的数据进行ReliefF改进后的算法分析，采用ReliefF改进后的算法与PCA算法结合，最终分析出最有可能的位点对疾病的关联。

1）由题目给出的数据，500个样本是患病的，500个是无病的数据，本文统计碱基在在患病与无病中的数量，然后把患病和无病中的碱基数量进行相减，如果相差过大，则说明此碱基对患病的影响的概率可能会比较大，如在位点rs1处的碱基对为TC、TT、CC，我们统计等位基因T在患病的样本中出现的次数和统计T的在无病的样本中出现的次数，用患病中T的次数减去无病中T的次数得到的结果，如果此结果相关过大，说明此碱基在患病中的影响可能会比较大，因此可以将它留下进行分析，去除影响小的碱基作为分析，然后进行筛选出一定的位点，再用剩余的位点进行MAF算法处理，进一步进行筛选位点工作；最小等位基因频率（Minor Allel Frequencey,MAF）是指某个位点少见等位基因的观察频率，GWAS中较小的MAF会使得统计效能降低，因此导致出现假阳性的结果。当MAF小于0.05时不作为全基因关联分析的研究对象，国际人类基因组单体型计划建议将MAF设置在0.05到0.5之间，根据一个SNP位点的MAF，可以筛选出一定数量的SNP位点，把未筛选的SNP进行关联分析处理。本文最终筛选后剩下111个SNP进行数据分析处理。

2）Relief算法是一种监督学习方法，通过对每个样本的寻找最近邻的样本来更新特征权重，是一种特征选择方法。Relief在更新每个特征权重时，考虑整个特征空间，具体做法是如果考虑某一个样本X的重要性特性时，先找到同一个类别最近邻的样式Y，然后找到一个不同类别的最近邻样本Z。如果某个特征在X和Y比较一致，但是在X和Z之间非常不同，那么这样的特征被认为是重要特征，对样本具有很强的区分度，会给这些特征赋一个较高的权重。Relief方法流程如下：

输入：M个训练样本X，每个训练样本N个特征，迭代次数为T（通常设为M）；

输出：每个特征的权重W[i][-1,1]；

初始化：对每个特征的权重初始化为0，W[i]=0；

for t=1 to T do

随机从数据集中选择一个样本X；

找到X的相同类别的最近邻样本和不同类别最近邻样本；

for i=1 to N do

 end for

end for

return(W).

对于SNP数据，基因型只有0,1,2三种取值，所以对每个特征位点i，样本X和Y的在特征i处的距离函数diff可以通过下面公式计算：



每一次迭代，每个特征的权重都得到更新，并且归一化为-1到1之间，输出是每个特征的权值，原始论文中，作者提出一个关联阈值，可以通过选择阈值大于等于的权重对应的特征来过滤特征，得到重要特征。Relief可以用来对特征进行排名。Relief算法同时进行全局和局部搜索，不像其他启发式搜索会陷入局最优。

3）Relief算法改进之后的算法为ReliefF算法，ReliefF算法Relief方法的改进版本，ReliefF算法的主要思想是根据属性在区分相互靠近的样本实例的能力上对属性的质量进行评价，首先随机地从训练数据中选择一个样本实例R，然后再训练数据中找出和样本实例R在同一类中的K个最近邻的样本实例，把这K个样本实例称为Nhits,然后再训练数据中分别找出和样本实例R不在同一个类中的K个最近邻的样本实例，把这些K个样本实例称作Nmisses;每个属性A的权重W[A]的更新依赖于随机选择的样本实例R、和R在同一类中的K个近邻NHits以及和R不在同一类中的若干K个近邻Nmisses,在属性权重的更新公式中所有NHits和NMisses的贡献是经过平均处理的。

基于对于Relief算法改进的主特征提取，基于前面介绍的原理，对改进Relief算法的过程如下：

(1) 去除无关特征变量。利用改进ReliefF算法删除原始特征中那些与分类不相关的特征。即当修改参数M（重复次数）和K（为最近邻的数目）时，有些特征的权重是始终保持不变的，这些特征就是与分类不相关的特征，可以把这些特征删除。经过处理后得到数据矩阵为



其中，表示一个n行p列的一个矩阵，代表n个样本实例，并且每个样本实例是p维的，可表示p个SNP位点；表示第j人特征变量在第i个样本实例上的观测值；代表第j个特征变量的观测向量；代表第i个样本实例的观测向量。

(2) 数据归一化处理。将矩阵转化为矩阵，变换公式为

其中，代表中第j列（第j列特征变量）的均值； 代表矩阵中第j个特征的标准差。

(3)进行主成分变换，先计算矩阵协方差矩阵，然后计算协方差矩阵的特征值和特征向量，特征值从大到小依次为，，…, 。

在近邻距离K=1时，各位点属性权重归一化后的值如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 位点 | 权重值 | 位点 | 权重值 |
| rs2273298 | 0.841 | rs556596 | 0.793 |
| rs7368252 | 0.828 | rs4391636 | 0.775 |
| rs2807345 | 0.821 | rs3795438 | 0.757 |
| rs1009806 | 0.821 | rs10864301 | 0.748 |
| rs10915035 | 0.812 | rs6429695 | 0.739 |
| rs4391636 | 0.811 | rs11580218 | 0.730 |
| rs2982376 | 0.802 | rs2801178 | 0.712 |

在近邻距离K=3时进行测试，前14位点属性权重归一化如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 位点 | 权重值 | 位点 | 权重值 |
| rs2273298 | 0.791 | rs11584631 | 0.67 |
| rs7368252 | 0.761 | rs4391636 | 0.651 |
| rs2807345 | 0.751 | rs3795438 | 0.649 |
| rs1009806 | 0.736 | rs10864301 | 0.642 |
| rs11588669 | 0.691 | rs1009806 | 0.637 |
| rs4391636 | 0.682 | rs11580218 | 0.632 |
| rs4646092 | 0.682 | rs2801178 | 0.627 |

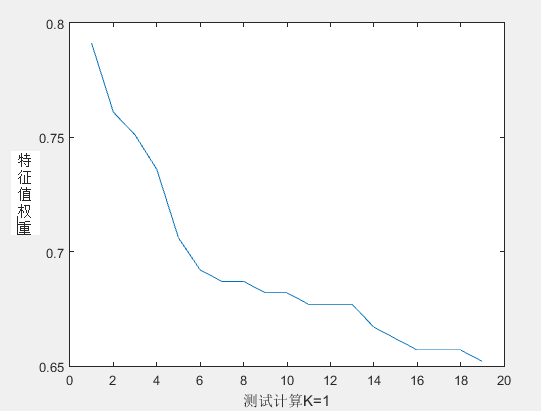
在近邻距离K=5时进行测试，前14位点属性权重归一化如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 位点 | 权重值 | 位点 | 权重值 |
| rs2273298 | 0.584 | rs556596 | 0.508 |
| rs7368252 | 0.563 | rs4391636 | 0.504 |
| rs2807345 | 0.556 | rs3795438 | 0.504 |
| rs1009806 | 0.546 | rs10864301 | 0.497 |
| rs10915035 | 0.525 | rs6429695 | 0.491 |
| rs4391636 | 0.515 | rs2801178 | 0.487 |
| rs11580218 | 0.511 | rs2473253 | 0.484 |

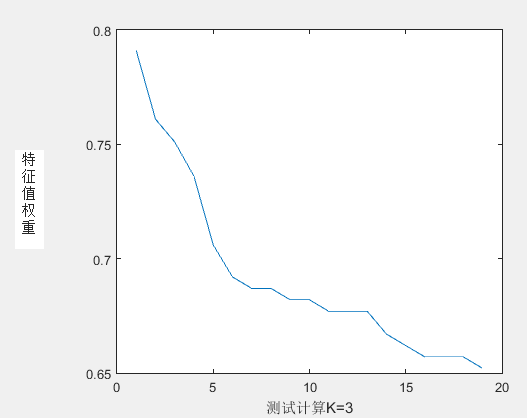
从以上图中可以得出在近邻距离K=1、K=3、K=5时，各位点的大小顺序趋势基本都是一致的。在K=1时从按照从小到大顺序排列，可知各个位点的权重关系如下：

rs2273298 < rs7368252 < rs2807345 < rs1009806 < rs10915035 < rs4391636 < rs4391636 < rs2982376 < rs556596，我们选定权重阀值为0.9，则其位点权重小于0.9的剔除。

从上面的特征权重可以看出，rs2273298大小是最主要的影响因素，说明患者的症状最先表现了位点rs2273298大小上，将直接导致患病的变化，其次是rs7368252、rs2807345以及rs1009806等，后几个位点权重大小接近，但是从多次计算规律来看，还是能够说明其中不同的重要程度，下面是着重对几个重要的位点属性进行分析。下面是测试近邻K=1时，位点的特征值权重图：



从上图中可以看到该属性权重大部分在0.75-0.8左右，是权重最大的一个属性。下面由近邻K=3时的权重分布：



从以上实验中,可以得知位点rs2273298、 rs7368252、rs2807345、rs1009806、 rs10915035、 rs4391636、rs4391636、rs2982376、 rs556596最有可能对疾病造成影响。

4）主成分分析是特征提取中很常见的一种变换方法，该方法通过使用变换后的几个主要的成分来代替原始的特征信息，变换后的主成分之间是相互独立的，并且每个主成分都是变换钱所有特征的一个线性组合，如果用变换后的所有主成分来代替原始的特征信息，一般没有信息的损失，因此，被广泛应用到模式匹配的相关领域中。但在很多情况下，通常选择少数的几个包含大部分原始特征信息，但可以使用更少的特征变量来代替原始较多的特征变量，这样可以大大减轻后续分类器的计算量，同时对提高分类器的性能也是有帮助的。

设,,……,为p个随机变量，记,令 为X的协方差矩阵，进行线性变换：



可以得到：



显然我们希望是,,…,的所有线性函数中方差最大的，此处限制为单位向量，即有，这样可使得达到最大，此时就称为第一主成分。如果第一主成分所包含的信息还不够多，不足以代表原始的p个变量，就要考虑使用、、等，一般来说，X的第i主成分指：和=0，k=1,2…,i-1下寻找,使得达到最大。

主成分分析结果：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 成分 | 初始特征值 | | | 提取平方和载入 | | |
| 合计 | 方差的% | 累积% | 合计 | 方差的% | 累积% |
| 1 | 5.024 | 50.235 | 50.23 | 5.024 | 50.235 | 50.235 |
| 2 | 2.080 | 20.35 | 70.5 | 2.080 | 20.799 | 71.034 |
| 3 | 0.735 | 7.385 | 77.9 | 0.735 | 7.386 | 78.3 |
| 4 | 0.686 | 6.857 | 86.7 | 0.686 | 6.56 | 84.83 |
| 5 | 0.376 | 3.767 | 91.4 |  |  |  |
| 6 | 0.303 | 3.024 | 94.4 |  |  |  |
| 7 | 0.286 | 2.78 | 96.6 |  |  |  |
| 8 | 0.224 | 2.56 | 98.1 |  |  |  |
| 9 | 0.211 | 1.047 | 99.1 |  |  |  |
| 10 | 0.083 | 0.830 | 100 |  |  |  |

这就是主成分分析的结果，表中第一列为10个成分；第二列为对应的“特征值”，表示所解释的方差的大小；第三列为对应的成分所包含的方差占总方差的百分比；第四列为累计的百分比。选择“特征值”大于1的成分作为主成分，这也是SPSS默认的选择。

在本题中成分1、2、3、4的特征值相对比较大，他们合计能解释大于90%的方差，所以我们可以提取1、2、3、4作为主成分，抓住了主要矛盾，其余成分包含的信息较少，故弃去。

综上所述，通过对位点和疾病的关联分析，我们去选最有可能引起疾病的位点为rs2273298、 rs7368252、rs2807345、rs1009806。

* + 1. 结论

由以上分析得，疾病发生的最有可能出现的位点为rs2273298、 rs7368252、rs2807345、rs1009806，在这些位点处出现的权重以及与疾病的关联性是最高的。

* + 1. 模型的进一步验证

卡方统计量是基于卡方分布的一种检验方法，根据数值频数值来构造统计量，是一种非参数检验方法。卡方值越大，越不符合；卡方值越小，偏差越小，越趋于符合，若两个值完全相等，卡方值为0，表明理论值完成符合。检验rs2273298、 rs7368252、rs2807345、rs1009806位点与疾病的关联关系。

(1).提出原假设：

由于总体分布为离散型，则假设具体为

.总体X的分布律为P{X=}=,i=1,2,…;

皮尔逊卡方检验统计量；

针对位点rs2273298，通过皮尔逊卡方检验，卡方值是28.835，自由度2，渐进显著水平为0.000，远小于5%。所以在是否患病的情况下，这三种碱基的数量有明显的差别。说明该位点为致病位点。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表：位点rs2273298与是否患病交叉表 | | | | | |
|  | | | 是否患病 | | 总计 |
| 患病 | 不患病 |
| rs2273298 | AA | 计数 | 222 | 305 | 527 |
| 预期计数 | 263.5 | 263.5 | 527.0 |
| AG | 计数 | 218 | 161 | 379 |
| 预期计数 | 189.5 | 189.5 | 379.0 |
| GG | 计数 | 60 | 34 | 94 |
| 预期计数 | 47.0 | 47.0 | 94.0 |
| 总计 | | 计数 | 500 | 500 | 1000 |
| 预期计数 | 500.0 | 500.0 | 1000.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 表：卡方检验 | | | |
|  | 值 | 自由度 | 渐近显著性 （双向） |
| 皮尔逊卡方 | 28.836a | 2 | .000 |
| 似然比(L) | 29.018 | 2 | .000 |
| 线性关联 | 27.379 | 1 | .000 |
| 有效个案数 | 1000 |  |  |
| a.0 个单元格 (0.0%) 具有的预期计数少于5。最小预期 计数为 47.00。 | | | | |

针对位点rs2807345，通过皮尔逊卡方检验，卡方值是13.198，自由度2，渐进显著水平为0.001，远小于5%。所以在是否患病的情况下，这三种碱基的数量有明显的差别。说明该位点为致病位点。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **rs2807345 \* 是否患病 交叉表** | | | | | |
|  | | | 是否患病 | | 总计 |
| 患病 | 不患病 |
| rs2807345 | CC | 计数 | 281 | 333 | 614 |
| 预期计数 | 307.0 | 307.0 | 614.0 |
| CT | 计数 | 179 | 145 | 324 |
| 预期计数 | 162.0 | 162.0 | 324.0 |
| TT | 计数 | 40 | 22 | 62 |
| 预期计数 | 31.0 | 31.0 | 62.0 |
| 总计 | | 计数 | 500 | 500 | 1000 |
| 预期计数 | 500.0 | 500.0 | 1000.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **卡方检验** | | | |
|  | 值 | 自由度 | 渐近显著性 （双向） |
| 皮尔逊卡方 | 13.198a | 2 | .001 |
| 似然比(L) | 13.285 | 2 | .001 |
| 线性关联 | 13.184 | 1 | .000 |
| 有效个案数 | 1000 |  |  |
| a. 0 个单元格 (0.0%) 具有的预期计数少于 5。最小预期计数为 31.00。 | | | |

同理可以得出rs7368252、rs2807345、rs1009806这三个位点的卡方检验分析，渐进显著水平为0.000，远小于5%。所以在是否患病的情况下，这三种碱基的数量有明显的差别。说明该位点为致病位点。

* 1. 问题3的分析求解
     1. 问题的分析

就目前所知，不同人的DNA序列99.9%是相同的，然而剩下的0.1%的不同导致了个体的多样性，它们导致了人们不同的属性与性格等差异。DNA上的差异有很大一部分是SNP。

对于问题二的求解，获得了每个位点对于疾病影响程度的权值。由于一个基因有多个位点，所以我们有如下分析，将一个基因理解为若干个位点组成的集合。将一个基因中的所有位点的权值累加起来，得到该基因的权值，该权值即表示该基因对疾病的影响程度。这样我们就得到了遗传疾病与该基因的关联性。选取权值大的一些基因，就找出了与疾病最有可能相关的基因。最终可以算得疾病最有可能的基因是gene\_ 55、gene\_ 217、gene\_ 293、gene\_ 30、gene\_ 83、gene\_ 113、gene\_ 144、gene\_ 162、gene\_ 169、gene\_ 22等19个基因。

* + 1. 数据的采集与处理

导入附件中的gene\_info文件夹里的gene\_0.dat到gene\_300.dat的基因序列文件。从而我们可以得到9445个位点所在300个基因的位置。

MATLAB代码如下：

temp=importdata('.\OrigionData\WeiDianIdByMAF.txt');

temp=temp';

numLocus\_MAF=size(temp,2);

IndexOfEachGenotype\_MAF(1,:)=1:1:numLocus\_MAF;

IndexOfEachGenotype\_MAF(2,:)=temp(:);

* + 1. 模型的建立与求解

问题3中，分析遗传疾病A与基因的关联关系，基因为若干个位点组成集合，遗传病与基因的关联性可以由包含的位点的全集或子集合表现出来。检查与疾病在统计学上的关联的因素，并以此作为潜在病因。从三百个基因文件中做统计，发现最多一个基因文件中有60个位点；利用geneotype.dat文件的第一行的位点名称对9445个位点进行编号，按照1,2,3,4……形式，因此每个基因文件中的位点可以用编号代替，将三百基因文件形成一个300\*60的矩阵的形式，300表示基因个数中60表示最大位点个数，矩阵里面数字为基因中位点的编号，对于很多基因没有60个位点的用0来填充。

在问题二中，我们算得出了所有位点的权值。定义起方程为



其中 表示第 个基因的权值；表示第n个位点是否在第*j*个基因中。



MATLAB代码如下：

weightOfEachGene\_MAF=zeros(2,numGene);

weightOfEachGene\_MAF(1,:)=1:1:numGene;

for j=1:numLocus\_MAF

weightOfEachGene\_MAF(2,mapOfGenotype\_cell{3,IndexOfEachGenotype\_MAF(2,j)})=weightOfEachGene\_MAF(2,mapOfGenotype\_cell{3,IndexOfEachGenotype\_MAF(2,j)})+1;

end

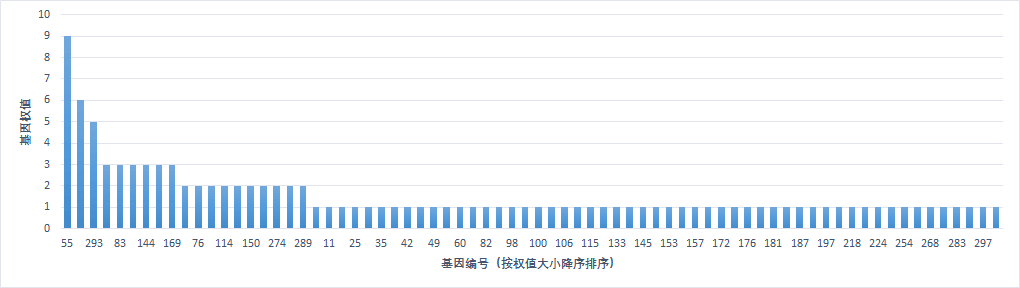
[result , idx]=sort(weightOfEachGene\_MAF(2,:),'descend');

numGene\_MAF=numGene-sum(weightOfEachGene\_MAF\_sorted\_result==0);

weightOfEachGene\_MAF\_sorted\_result=result(1:numGene\_MAF);

weightOfEachGene\_MAF\_sorted\_index=idx(1:numGene\_MAF);

将所有基因以基因权值为依据，按照从大到小的顺序排序，据此计算出300个基因中，与111个致病位点有关的72个候选致病基因，并计算出72个候选致病基因的权值，将这些基因以基因权值为依据，按照从大到小的顺序排序。我们发现，即使是权值最大的基因，对疾病的贡献率也仅有0.087%。因此，我们需要计算累计贡献率，并将累计贡献率在50%前的所有基因列入与疾病有关的基因集合中。绘制出基因权值排序柱状图如下：



从图中我们可以发现前19个基因的权重相对来说是比较大的，后面的基因都是处于一种趋势发展；所以我们认为前19个基因可能是造成疾病的原因。

基因的权重值如下图：

|  |  |  |  |
| --- | --- | --- | --- |
| 基因 | 权重值 | 基因 | 权重值 |
| gene\_ 55 | 9 | gene\_ 22 | 2 |
| gene\_ 217 | 6 | gene\_ 76 | 2 |
| gene\_ 293 | 5 | gene\_ 102 | 2 |
| gene\_ 30 | 3 | gene\_ 114 | 2 |
| gene\_ 83 | 3 | gene\_ 149 | 2 |
| gene\_ 113 | 3 | gene\_ 150 | 2 |
| gene\_ 144 | 3 | gene\_ 241 | 2 |
| gene\_ 162 | 3 | gene\_ 274 | 2 |
| gene\_ 169 | 3 | gene\_ 280 | 2 |

* + 1. 结论

通过计算，我们得到的与疾病有关的基因集合有19个。

* + 1. 模型的进一步验证

问题三进行卡方检验，针对基因gene\_ 55，通过皮尔逊卡方检验，卡方值是28.835，自由度2，渐进显著水平为0.015，远小于5%。所以在是否患病的情况下，该基因里面的位点数量有明显的差别。说明该基因为致病基因。同理可得其它基因的卡方检验，得到它们的渐进显著水平都为小于5%，因此在是否引起疾病的情况下，该基因里面的位点数量有明显的差别，所检验的基因为致病的基因。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Gene\_55\* 是否患病 交叉表** | | | | | |
|  | | | 是否患病 | | 总计 |
| 患病 | 不患病 |
| gene\_ 55 | rs7368252 | 计数 | 158 | 160 | 298 |
| 预期计数 | 148.0 | 150.0 | 298.0 |
| rs7366321 | 计数 | 251 | 261 | 513 |
| 预期计数 | 246.5 | 256.5 | 513.0 |
| rs11801490 | 计数 | 67 | 122 | 189 |
| 预期计数 | 84.5 | 104.5 | 189.0 |
| 总计 | | 计数 | 500 | 500 | 1000 |
| 预期计数 | 500.0 | 500.0 | 1000.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **卡方检验** | | | |
|  | 值 | 自由度 | 渐近显著性 （双向） |
| 皮尔逊卡方 | 18.435a | 2 | .015 |
| 似然比(L) | 8.474 | 2 | .014 |
| 线性关联 | 6.831 | 1 | .009 |
| 有效个案数 | 1000 |  |  |
| a. 0 个单元格 (0.0%) 具有的预期计数少于 5。最小预期计数为 94.50。 | | | |

* 1. 问题4的分析与求解
     1. 问题的分析与求解

问题四中讨论10个相关性状的信息与其相关联的位点，给出的是10个性状，然后每个性状有1000样本，考虑到10个性状和位点之间的关系，可以转化先考虑单个性状和位点之间的关系，也就是转化为了当初的疾病和位点之间的关系，这里的不同之处在于当初疾病前500个是0，后500个是1，这里性状1000个样本的0和1是随机出现的，没有规律，先单个用算法计算出与每个性状相关的位点集合，最后统计存在这10个位点集合中共同位点作为整体相关联的性状位点。

* + 1. 数据的采集与处理

采用题目中给定的genotype.dat与multi\_phenos.txt文件数据，从multi\_phenos.txt文件中1000个样本中10个性状与位点的关系，分析此两文件数据的关联关系，从而分析出疾病最有可能是哪些位点造成的。

MATLAB核心代码实现如下：

%% 筛选出十个权值里的位点的重复项

mark\_Q4=zeros(numSamples,10);

repeatted=zeros(1,numLocus);

for i=1:1:numSamples

for j=1:1:10

if IndexOfEachGenotype\_Q4(i,j)==0

break;

end

repeatted(1,IndexOfEachGenotype\_Q4(i,j))= repeatted(1,IndexOfEachGenotype\_Q4(i,j))+1;

end

end

%% 计算重复项的权值

[repeatted\_result,repeatted\_index]=sort(repeatted,'descend');

* + 1. 结论

通过求解得出与10个性状相关的位点为：

rs10909903,rs6686587,rs311469,rs3128326,rs608203,  
rs10799145,rs10492987,rs7545115,rs2501401,rs4314833,  
rs351619,rs770706,rs6678618,rs11247946 。

1. 模型评价与推广

本文建议的疾病与位点以及基因的理论基本模型合理，基本符合实际情况，可广泛应用到疾病的分析求解问题中，在分析位点、基因和疾病的问题时，有独特的用途。但文章中的一些参数是通过理论计算得到的，对于不同的疾病和位点要做不同的研究。

1. 参考文献

[1] 黎成. 基于随机森林和ReliefF的致病SNP识别方法[D]. 西安：西安电子科技大学，2014.

[2]. 吴水秀，曾庆鹏等. 基于改进ReliefF算法的主成分特征提取. 计算机工程，2008.

[3] Md.Mamun Monir,Zhu Jun等.数量性状位点定位和全基因组关联分析的方法与软件综述.浙江大学学报，2014.

[4] 陈海林. 网络医学中若干关联问题的计算分析[M]. 中南大学信息科学与工程学院. 2013.

[5] 彭倩倩. 群体病例对照研究设计的整体基因关联分析统计推断方法研究[D]. 山东大学. 2009.

[6] 胡艳玲. 复杂性状与基因组多位点的关联分析方法研究[M]. 上海交通大学. 2009.

[7] 杨小红. 关联分析的一般思路-存在问题及发展趋势. 中国农业大学国家玉米改良中心. 2011.

[8] 丁小军. 大规模生物数据信息挖掘算法设计[M]. 中南大学信息科学与工程学院. 2014.

[9] 葛鹏程. 生物学数据挖掘算法的设计与实现[D]. 哈尔滨工业大学, 2005.

[10] 张平. 生物数据多层关联规则挖掘算法的优化设计[D]. 华中科技大学, 2007.

1. 附录

核心代码如下：

(1)降维

**public** **void** CalMAF(**float** yuzhi,String path,String pathid)

{

init(path);

Sample head=al.get(0);

String [] headName=head.getGenetype().split("\\s++");

Sample tail=al.get(500);

String [] tailName=head.getGenetype().split("\\s++");

**float** MAFa;

**for**(**int** i=0;i<N;i++)//位点

{

**int** rate1=0;

**int** rate2=0;

GeneMAF gm=**new** GeneMAF();

//识别标识

String gene=headName[i];

**char** tag1=gene.charAt(0);

rate1++;

**if**(gene.charAt(1)==tag1)

rate1++;

**for**(**int** j=1;j<500;j++)

{

Sample sample=al.get(j);

gene=sample.getGenetype().split("\\s++")[i];

**if**(gene.charAt(0)==tag1)

rate1++;

**if**(gene.charAt(1)==tag1)

rate1++;

}

String tai=tailName[i];

**char** tag2=tai.charAt(0);

rate2++;

**if**(gene.charAt(1)==tag2)

rate2++;

**for**(**int** k=501;k<1000;k++)

{

Sample sample=al.get(k);

gene=sample.getGenetype().split("\\s++")[i];

**if**(gene.charAt(0)==tag2)

rate2++;

**if**(gene.charAt(1)==tag2)

rate2++;

}

//筛选数据

**if**(tag1==tag2) //tag1和tag2表示位点相同

{

MAFa=Math.*abs*((rate1-rate2)/(**float**)1000);

**if**(MAFa>yuzhi)

{

WeiDianId.add(i);

System.*out*.println(MAFa);

}

}

//表示位点不相同

**else**

{

MAFa=Math.*abs*((1000-rate1-rate2)/(**float**)1000);

**if**(MAFa>yuzhi)

{

WeiDianId.add(i);

System.*out*.println(MAFa);

}

}

}

writeWeidianId(pathid);

System.*out*.println("剩余位点:"+ WeiDianId.size());

}

(2) 获取同一类别的近邻id

**public** ArrayList<Neighbor> getSameNearstId(Sample training)

{

ArrayList<Neighbor> Neighbors=**new** ArrayList<Neighbor>();

**int** id=training.getId()-1;

// System.out.println("id"+id);

**if**(training.isTag())//表示训练集是有病的状态

{

Neighbors = lastCase(training, Neighbors, id);

//排序按照 weight相关性大小排序

Collections.*sort*(Neighbors,comparator);

// paixu(Neighbors);

}

**else**

{

Neighbors = forecase(training, Neighbors, id);

//排序按照 weight相关性大小排序

Collections.*sort*(Neighbors,comparator);

// paixu(Neighbors);

// System.out.println("Neighbors:"+Neighbors.size());

}

**return** Neighbors;

}

(3)获取在不同类别中的近邻idsss

**public** ArrayList<Neighbor> getDistinctNearstId(Sample training)

{

ArrayList<Neighbor> Neighbors=**new** ArrayList<Neighbor>();

**int** id=training.getId()-1;

// int sameGene=0;//相同的基因组

**if**(!training.isTag())//表示训练集是无病的状态

{

Neighbors = lastCase(training, Neighbors, id);

//排序按照 weight相关性大小排序

Collections.*sort*(Neighbors,comparator);

// paixu(Neighbors);

}

//有病的状态

**else**

{

Neighbors = forecase(training, Neighbors, id);

//排序按照 weight相关性大小排序

Collections.*sort*(Neighbors,comparator);

// paixu(Neighbors);

}

**return** Neighbors;

}

（4）ReliefF算法

//releifF算法主程序,path是路径，k是控制测试的近邻个数

**public** **void** mainProcedure(String path,**int** k,String outpath,String pathimportid)

{

init(path);

readImportweidian(pathimportid);

//初始化一个位点的数组

**double** weight[]=**new** **double**[WeiDianId.size()];

**float** diffsame;

**float** diffdistinct;

**for**(**int** i=0;i<WeiDianId.size();i++)

{

weight[i]=0;

}

**int** e=10;

**int** s=0;

**int** id;

**int** importantid;

**for**(**int** j=0;j<1000;j++)//迭代次数

{

//int id=(int) Math.round(Math.random()\* 999);

//获取样本

Sample test=al.get(j);

ArrayList<Neighbor> sameNeighbors=getSameNearstId(test);

ArrayList<Neighbor> distinctNeighbors=getDistinctNearstId(test);

**for**(**int** k1=0;k1<WeiDianId.size();k1++)

{

importantid=WeiDianId.get(k1);

**for**(**int** f=0;f<k;f++)

{

diffsame=diffValue(j, sameNeighbors.get(f).getId(),importantid);

diffdistinct=diffValue(j,distinctNeighbors.get(f).getId(),importantid);

weight[k1]=weight[k1]+(diffdistinct-diffsame)/(**double**)(N\*T);

}

}

System.*out*.println(j);

}

s writeTofile(outpath, weight);

**for**(**int** t=0;t<10;t++)

{

System.*out*.println(SNPname[WeiDianId.get(t)]+":"+weight[t]);

}

}

（5）在相同类型的样本

private ArrayList<Neighbor> SameCase(Sample training, ArrayList<Neighbor> Neighbors,

int id)

{

int tag=training.isTag();

int sameGene=0;//相同的基因组

String []trainingSplit=training.getGenetype().split("\\s+");

for(int i=0;i<1000;i++)

{

if(i!=id&&biaoshi.get(i)==tag)

{

sameGene=0;

Neighbor neighbor=new Neighbor();

//获取样本

Sample sample=al.get(i);

String []sampleSplit=sample.getGenetype().split("\\s+");

// System.out.println("samplesplit:"+sampleSplit.length+"id"+i);

for(int j=0;j<sampleSplit.length;j++)

{

if(trainingSplit[j].equals(sampleSplit[j]))

{

sameGene++;

}

}

//将id和权值加入到邻居中

neighbor.setId(i);

neighbor.setWeight(sameGene/(float)N);

Neighbors.add(neighbor);

}

}

return Neighbors;

}

//在不同类型的样本

private ArrayList<Neighbor> DifferentCase(Sample training, ArrayList<Neighbor> Neighbors,

int id)

{

int tag=training.isTag();

int sameGene=0;//相同的基因组

String []trainingSplit=training.getGenetype().split("\\s+");

//System.out.println("trainingSplit:"+trainingSplit.length);

for(int i=0;i<1000;i++)

{

if(i!=id&&biaoshi.get(i)!=tag)

{

sameGene=0;

Neighbor neighbor=new Neighbor();

//获取样本

Sample sample=al.get(i);

String []sampleSplit=sample.getGenetype().split("\\s+");

// System.out.println("samplesplit:"+sampleSplit.length+"id"+i);

for(int j=0;j<sampleSplit.length;j++)

{

if(trainingSplit[j].equals(sampleSplit[j]))

{

sameGene++;

}

}

//将id和权值加入到邻居中

neighbor.setId(i);

neighbor.setWeight(sameGene/(float)N);

Neighbors.add(neighbor);

}

}

return Neighbors;

}