

上海科技大学 本科生
(2018年-2019第1学期)
《数学建模》课程小论文

姓 名	郭瑞昊，马荣泽，张上，张博炜
学 号	34562686, 50816956, 23005066, 99807733
论文题目	基于统计学习的治病基因位点预测分析方法探究
参考文献 题目	【1】 基因水平的关联分析方法 【2】 精通机器学习：基于 R（第二版） 【3】 July.支持向量机通俗导论（理解 SVM 的三层境界）
最便捷联系电话	13918054286
Email	guorh@shanghaitech.edu.cn
指导教师	肖柳青

基于统计学习的治病基因位点预测分析方法探究

郭瑞昊，马荣泽，张上，张博炜

【摘要】：在人类疾病当中，有很大一部分是遗传疾病与遗传信息紧密相关。DNA 是遗传信息的载体，其多态性与疾病是有着紧密联系的。全基因组关联研究已成为研究基因与疾病关系的有效手段。本文将统计学习的方法应用于全基因组关联研究，研究判断基因与疾病之间的关系，尝试提供一种新的根据基因预测患病可能的有效方法。

【关键词】 基因分析 统计学习 K 最邻近 支持向量机

基于统计学习的治病基因位点预测分析方法探究

【摘要】：在人类疾病当中，有很大一部分是遗传疾病与遗传信息紧密相关。DNA 是遗传信息的载体，其多态性与疾病是有着紧密联系的。全基因组关联研究已成为研究基因与疾病关系的有效手段。本文将统计学习的方法应用于全基因组关联研究，研究判断基因与疾病之间的关系，尝试提供一种新的根据基因预测患病可能的有效方法。

【关键词】 基因分析 统计学习 K 最邻近 支持向量机

一、问题描述

人类的每条染色体都带有一个 DNA 分子，DNA 分子中携带有人类的几乎全部的遗传信息。DNA 分子是由含有腺嘌呤（A）、鸟嘌呤（G）、胞嘧啶（C）、胸腺嘧啶（T）这四种碱基的脱氧核糖核苷酸组成的双螺旋长链大分子。人类细胞内共存在有超过 30 亿个碱基对，决定了人类的发育进程。在这些碱基对中，有一些特定位点在不同的人之间是有区别的，这称为 DNA 的多态性。大量的研究表明人类的某些疾病与 DNA 的多态性位点是有着紧密联系的。

目前，利用患病者和健康者的 DNA 位点之间的区别的全基因组关联研究（Genome wide association study, GWAS）已在遗传学相关的疾病研究中得到了广泛的应用。包括精神分裂症、冠心病等多基因复杂性状的疾病适合利用 GWAS 方法来确定致病位点的范围，便于预防和治疗相关疾病。^[1]

目前，GWAS 方法往往只对单一的 SNP 位点进行分析。其一般首先会利用关联性分析（趋势性检验或等位基因检验）的方法，计算出每一个 SNP 与疾病的关联效应（P 值越大，关联效应越弱），之后利用其它方法根据 P 值得不同判断 DNA 的区别与该种疾病发生之间的关系。采用的方法包括了最显著 SNP 法，次显著 SNP 法以及 simes 法。这些方法大多只将 P 值最小或次小的 SNP 位点作为疾病产生因素，这就使得其忽略了其他结果可能存在“假阴性”的情况。对于复杂的多基因遗传病来说，这种方法仅能找到最相关的一个 SNP 位点，而其他的位点影响就被忽略了，很难对判断一个人的基因对于疾病是否发生的影响。

目前统计学习的运用已经相当广泛，常见的如 K 最临近算法(KNN)，支持向量机(SVM)等，因此我们提出设想：如果将统计学习中的方法直接运用到基因分析的研究中是否可以改进这个问题？

二、问题分析

大量研究表明，人体的许多表型性状差异以及对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。

近年来，研究人员大都采用全基因组的方法来确定致病位点或致病基因，具体做法是：招募大量志愿者（样本），包括具有某种遗传病的人和健康的人，通常用 1 表示病人，0 表示健康者。对每个样本，采用碱基 (A, T, C, G) 的编码方式来获取每个位点的信息 (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)；研究人员可以通过对样本的健康状况和位点编码的对比分析来确定致病位点，从而发现遗传病或性状的遗传机理。

我们从往届数学建模的题库中找一份数据，其中包含可能致病的位点编码信息 (genotype.dat) 以及样本的患病信息 (phenotype.txt)。由于基因的测序结果一般是以每个位点的碱基 (A, G, C, T) 的方式描述的，所以可以将其转化为数值编码的方式，运用统计学习的方法，可以对编码信息和患病信息进行分析，找出该种疾病最有可能的一个或几个致病位点，提供判断是否患病的预测模型。

三、问题假设及数据模型

对获取到的数据进行预处理，将基因信息和患病信息以数值编码的形式重新存储。

在我们找到的数据中 genotype.dat 存储有 1000 组基因位点信息，包含 9445 个基因位点上上的编码信息。如图 (1) 所示，1 号样本在 rs3094315 位点上的基因编码为“TT”，在 rs3131972 位点上的基因编码为“CT”，在 rs3131969 位点上编码为“CC”等等。

	rs3094315	rs3131972	rs3131969	rs1048488	rs12562034
1	TT	CT	CC	TC	AA
2	CT	TC	GG	AG	AA
3	CC	AA	CC	GG	CC
4	AA	TT	TT	GG	CC
5	TT	CT	CC	TC	AA
6	CT	TC	GG	AG	AA
7	CC	AA	CC	GG	CC
8	AA	TT	TT	GG	CC
9	TT	CT	CC	TC	AA
10	CT	TC	GG	AG	AA
11	CC	AA	CC	GG	CC
12	AA	TT	TT	GG	CC
13	TT	CT	CC	TC	AA
14	CT	TC	GG	AG	AA
15	CC	AA	CC	GG	CC
16	AA	TT	TT	GG	CC
17	TT	CT	CC	TC	AA
18	CT	TC	GG	AG	AA
19	CC	AA	CC	GG	CC
20	AA	TT	TT	GG	CC
21	TT	CT	CC	TC	AA
22	CT	TC	GG	AG	AA
23	CC	AA	CC	GG	CC
24	AA	TT	TT	GG	CC
25	TT	CT	CC	TC	AA
26	CT	TC	GG	AG	AA
27	CC	AA	CC	GG	CC
28	AA	TT	TT	GG	CC

图 1 基因位点及对应位点的编码信息

由背景知识可知一个位点上的 DNA 碱基对只有 AT, CG 两种组合方式, 因此推测数据中的位点编码信息对应相邻的两个碱基对。另外, 在 phenotype.txt 中以一列的形式存储了对应 1000 组数据的类型 (“1” 表示病人, “0” 表示健康)。

	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0

图 (2) 患病状态信息

首先将 genotype.dat 中的“TT”，“TC” ……等共 19 种碱基信息对应为 0 至 18 的整数（除 A，T，C，G 四种碱基相互配对形成的 16 种以外，在 9445 个位点中的第 6070 和第 8472 这两个位点的基因信息中出现“II”，“ID”，“DD”三种基因型,存在一些稀有碱基），便于进一步的数据分析，具体对应关系如表（1）所示。经过对文件中的字符进行替换后，将得到的数据保存为 gebotype_int.dat，如图（3）所示。

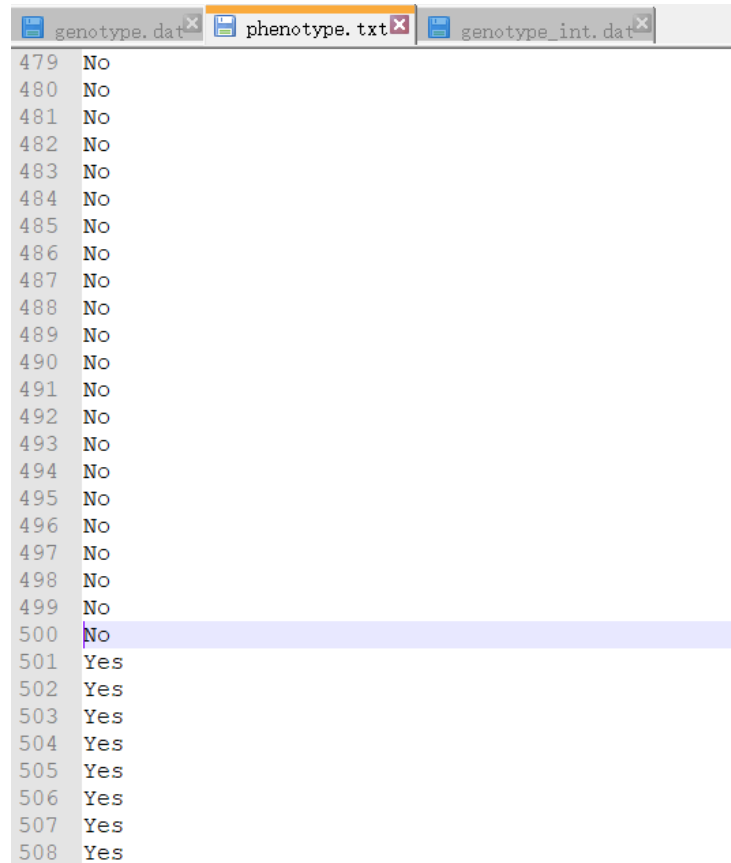
AA	AT	AC	AG
0	1	2	3
TA	TT	TC	TG
4	5	6	7
CA	CT	CC	CG
8	9	10	11
GA	GT	GC	GG
12	13	14	15
II	ID	DD	
16	17	18	

表(1) 碱基型号与数值的对应关系

	rs3094315	rs3131972	rs3131969	rs1048488	rs1256203
1	5	9	10	6	0
2	0	0	10	15	10
3	0	0	6	10	5
4	15	5	15	10	15
5	10	10	15	10	10
6	15	3	15	10	15
7	10	15	10	10	15
8	12	0	3	9	13
9	12	0	3	9	13
10	10	6	12	0	3
11	10	6	12	0	3
12	10	6	12	0	3
13	10	6	12	0	3
14	10	6	12	0	3
15	10	6	12	0	3
16	10	6	12	0	3
17	10	6	12	0	3
18	10	6	12	0	3
19	10	6	12	0	3
20	10	6	12	0	3
21	10	6	12	0	3
22	10	6	12	0	3
23	10	6	12	0	3
24	10	6	12	0	3
25	10	6	12	0	3
26	10	6	12	0	3
27	10	6	12	0	3
28	10	6	12	0	3

图（3）将基因型信息替换为数字后得到数据 genotype_int.dat

同时,将文件 phenotype.txt 中用“0”,“1”表示的患病数据分别替换为“No”和“Yes”,便于进行分类。



The screenshot shows a text editor with three tabs: 'genotype.dat', 'phenotype.txt', and 'genotype_int.dat'. The 'phenotype.txt' tab is active and displays a list of 1000 samples. Samples 479 through 499 are labeled 'No', sample 500 is highlighted in blue and labeled 'No', samples 501 through 508 are labeled 'Yes', and samples 509 through 1000 are not visible. The text is left-aligned with sample IDs in the first column and the disease status in the second column.

479	No
480	No
481	No
482	No
483	No
484	No
485	No
486	No
487	No
488	No
489	No
490	No
491	No
492	No
493	No
494	No
495	No
496	No
497	No
498	No
499	No
500	No
501	Yes
502	Yes
503	Yes
504	Yes
505	Yes
506	Yes
507	Yes
508	Yes

图（4）将文件 phenotype.txt 中“0”，“1”数值分别替换为“No”和“Yes”

经过转换后数据呈现为 1000 个样本点,每个样本具有 9446 个特征。其中 9445 个为整数值,1 个为“Yes”和“No”的二值变量。从而将任务转化为统计学习中的分类问题。

四、模型建立和评估

使用 R 语言对数据进行分析,使用的版本为 R x64 3.5.2

加载对应的程序包:

```
> library(class)
> library(kknn)
> library(e1071)
> library(caret)
```

从处理后的 genotype_int.dat 中读取数据，检查后确认确实为 1000 组数据，变量数量为 9445 个。

```
> geno = read.table("genotype_int.dat", TRUE)
> str(geno)
'data.frame':   1000 obs. of  9445 variables:
```

同样，从 pheno.dat 读入数据并检查。

```
> pheno = read.table("phenotype.txt")
> str(pheno)
'data.frame':   1000 obs. of  1 variable:
 $ V1: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

将 genotype.dat 中的数据归一化后将 phenotype 中的类型信息合并如 genotype 中，确认合并后的变量数量为 9446。

```
> geno <- data.frame(scale(geno[,1:9445]))
> geno$type <- pheno$V1
> str(geno)
'data.frame':   1000 obs. of  9446 variables:
```

以 70/30 为划分比例随机将数据分为训练集和测试集。

```
> set.seed(123)
> ind <- sample(2, nrow(geno), replace = TRUE, prob = c(0.7, 0.3))
> train <- geno[ind == 1, ]
> str(train)
'data.frame':   705 obs. of  9446 variables:
> test <- geno[ind == 2, ]
> str(test)
'data.frame':   295 obs. of  9446 variables:
```

3.1 KNN 建模

KNN处理分类问题的方法是找最近的点（最近邻）来确定正确的分类。k的作用是确定算法应该检查多少个近邻，如果k = 5，算法将检查5个最近的点。这种方法的缺点是，所有5个点在算法中都被赋予相同的权重——尽管它们在学习过程中的相关性不高。^[2]

首先直接使用 KNN 算法对数据进行训练

```
> grid1 <- expand.grid(k = seq(2, 20, by = 1))
> control <- trainControl(method = "cv")
> set.seed(502)
> knn.train <- train(type ~ ., data = train,
+ method = "knn",
+ trControl = control,
+ tuneGrid = grid1)
> knn.train
k-Nearest Neighbors
```

705 samples

9445 predictors

2 classes: 'No', 'Yes'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 634, 634, 634, 634, 635, 634, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
2	0.5078450	0.014549510
3	0.5034564	0.005901393
4	0.4904768	-0.019880811
5	0.5063342	0.011682023
6	0.5050477	0.009077702
7	0.4893535	-0.022743411
8	0.4921905	-0.016809112
9	0.4950281	-0.011133797
10	0.4794760	-0.041646345

11	0.5091132	0.017146179
12	0.4949873	-0.011511915
13	0.5034392	0.005396547
14	0.4906625	-0.019702991
15	0.5047666	0.008124722
16	0.5146648	0.027931667
17	0.5089511	0.016045910
18	0.4891718	-0.022930687
19	0.5076243	0.013824510
20	0.4976852	-0.005857957

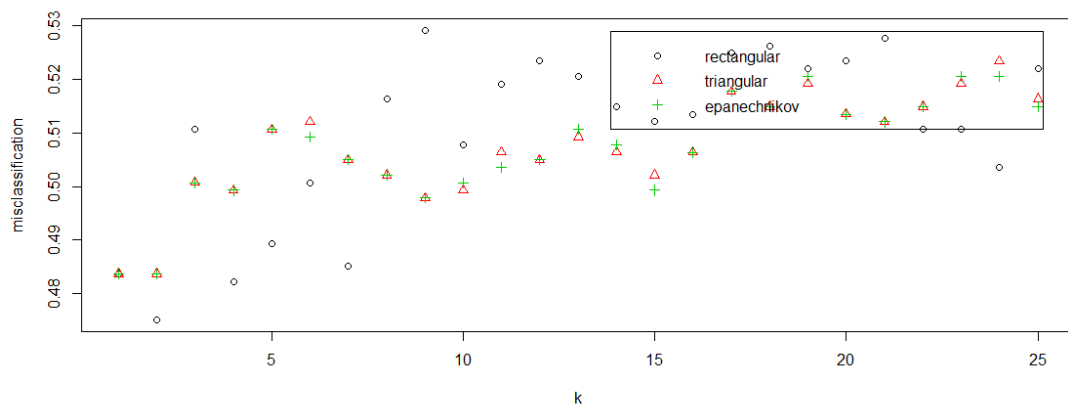
Accuracy was used to select the optimal model using the largest value.

The final value used for the model was $k = 16$.

得到的模型正确率均不理想。

进一步使用不同的距离加权方式，使用 `kknn` 函数进行训练，并将结果作图。

```
> set.seed(123)
> kknn.train <- train.kknn(type ~ ., data = train, kmax = 25,
+ distance = 2,
+ kernel = c("rectangular", "triangular", "epanechnikov"))
> plot(kknn.train)
```



图（5） kknn 训练结果

```
> kknn.train
```

Call:

```
train.kknn(formula = type ~ ., data = train, kmax = 25, distance = 2, kernel =  
c("rectangular", "triangular", "epanechnikov"))
```

Type of response variable: nominal

Minimal misclassification: 0.4751773

Best kernel: rectangular

Best k: 2

最优的模型为不加权的方式 ("rectangular")。从上面的数据可以看出，给距离加权不能提高模型在训练集上的正确率。从下面的代码可以看出，它同样没有提高测试集上的正确率：

```
> kknn.pred <- predict(kknn.train, newdata = test)
```

```
> table(kknn.pred, test$type)
```

kknn.pred No Yes

No 66 69

Yes 77 83

```
> (66+83)/295
```

```
[1] 0.5050847
```

3.2 主成分分析

直接使用 KNN 进行训练并不成功，推测可能的原因是变量过多，所以想到先对数据进行主成分分析（PCA），从而减少数据维度。

```
> pca <- prcomp(geno[, 1:9445], scale = T)
```

```
> summary(pca)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	6.32887	6.18630	6.13717	6.03185	5.98566	5.93086
Proportion of Variance	0.00424	0.00405	0.00399	0.00385	0.00379	0.00372
Cumulative Proportion	0.00424	0.00829	0.01228	0.01613	0.01993	0.02365

PC999 PC1000

Standard deviation 1.88179 2.635e-14

Proportion of Variance 0.00037 0.000e+00

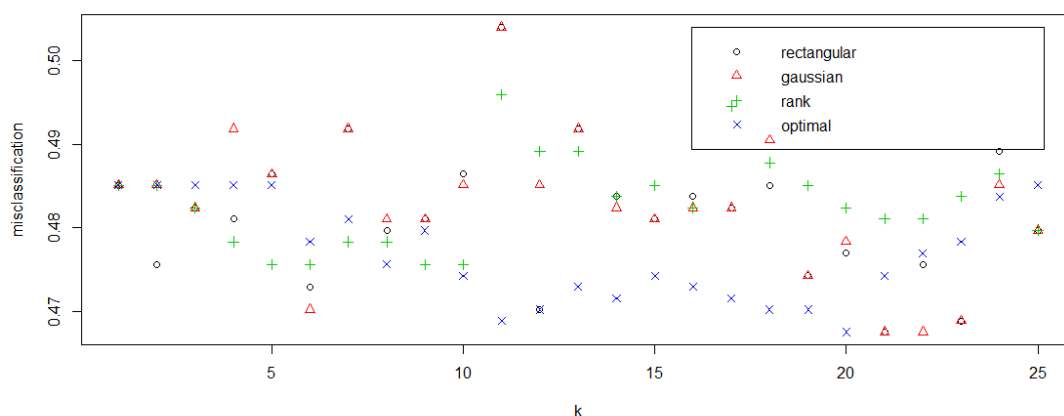
Cumulative Proportion 1.00000 1.000e+00

从截取的结果中看出 PCA 的结果也并不理想，因此保留了前 999 个变量重新生成训练集和测试集：

```
> geno.pca <- data.frame(pca$x[, 1:999], type = pheno$V1, scale = T)
> set.seed(123)
> ind <- sample(2, nrow(geno.pca), replace = TRUE, prob = c(0.7, 0.3))
> train2 <- geno.pca[ind == 1, ]
> test2 <- geno.pca[ind == 2, ]
```

重新构建 KNN 模型并使用绝对值距离以及另外三种加权方式 ("rectangular", "gaussian", "rank", "optimal"):

```
> kknn.train2 <- train.kknn(type ~ ., data = train2, kmax = 25,
+ distance = 1,
+ kernel = c("rectangular", "gaussian", "rank", "optimal"))
> plot(kknn.train)
```



图（6） 经过 PCA 后数据的 kknn 训练结果

```
> kknn.train2
```

Call:

```
train.kknn(formula = type ~ ., data = train2, kmax = 25, distance = 1, kernel =
```

```
c("rectangular", "gaussian", "rank", "optimal"))
```

Type of response variable: nominal

Minimal misclassification: 0.4674797

Best kernel: rectangular

Best k: 21

```
> kknn.pred2 <- predict(kknn.train2, newdata = test2)
```

```
> table(kknn.pred2, test2$type)
```

kknn.pred2 No Yes

No 69 75

Yes 45 73

```
> (69+73)/262
```

```
[1] 0.5419847
```

这一次正确率终于有了一定的提高，但仍然远没有达到可以接受的程度。

3.3 支持向量机

支持向量机 (support vector machine, SVM) 是机器学习里的一种分类算法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，即支持向量机的学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。^[3]

SVM 中的一个重要问题是处理非线性模型的能力，非线性模型的输入特征带有二次项或更高阶的多项式。在 SVM 中，这种处理被称为核技巧。核函数的巧妙之处在于，它将特征到高维空间的转换进行了数学上的简化，不需要在高维空间中显式地创建特征。这样做的好处是，在建立高维非线性空间和决策边界的同时，还能保持最优问题的计算有效性。核函数不用将特征转换到高维空间即可计算特征在高维空间中的内积。^[2]

因为在之前的步骤中已经将数据归一化并进行了主成分分析，此时可以直接将划分好的训练集用来进行 SVM 训练。

```
> linear.tune <- tune.svm(type ~., data = train2, kernel = "linear",
```

```
+cost = c(0.001,0.01,0.1,1,5))
```

```
> summary(linear.tune)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:

cost

0.01

- best performance: 0.5133803

- Detailed performance results:

	cost	error	dispersion
1	0.001	0.5248089	0.05424600
.....		
5	5.000	0.5133803	0.05014178

首先使用的是线性核函数进行训练，得到的模型与随机分类几乎没有区别。
使用多项式核函数以后结果有了略微的提升：

```
> poly.tune <- tune.svm(type~ ., data = train2,  
+ kernel = "polynomial",  
+ degree = c(3, 4, 5),  
+ coef0 = c(0.1, 0.5, 1, 2, 3, 4))  
> summary(poly.tune)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:

degree coef0

3 0.1

- best performance: 0.4937425

- Detailed performance results:

	degree	coef0	error	dispersion
1	3	0.1	0.4937425	0.02881840
.....			
18	5	4.0	0.5051107	0.05184351

而使用径向基核函数进行测试时，则甚至还不如线性核函数下的最优结果：

```
> set.seed(123)
> rbf.tune <- tune.svm(type~ ., data = train2,
+ kernel = "radial",
+ gamma = c(0.1, 0.5, 1, 2))
> summary(rbf.tune)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:

gamma
0.1

- best performance: 0.520503

- Detailed performance results:

	gamma	error	dispersion
1	0.1	0.5205030	0.04497875
.....		
4	2.0	0.5247284	0.04698384

使用 SVM 建模的最优结果在同时改变两个参数 gamma 和核系数 coef0 时得到：

```
> set.seed(123)
> sigmoid.tune <- tune.svm(y ~ ., data = train,
+ kernel = "sigmoid",
+ gamma = c(0.1, 0.5, 1, 2, 3, 4),
```

```
+ coef0 = c(0.1, 0.5, 1, 2, 3, 4))
```

```
> summary(sigmoid.tune)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:

gamma coef0

1 4

- best performance: 0.4592555

- Detailed performance results:

	gamma	coef0		error	dispersion
1	0.1	0.1	0.5108048	0.07662396	
.....				
36	4.0	4.0	0.5078873	0.08340287	

```
> best.sigmoid <- sigmoid.tune$best.model
```

```
> sigmoid.test <- predict(best.sigmoid, newdata = test)
```

```
> table(sigmoid.test, test$y)
```

```
sigmoid.test No Yes
```

```
      No    23   15
```

```
      Yes 119 138
```

```
> (23+138)/295
```

```
[1] 0.5457627
```

使用测试集对所得的最优模型进行验证，得到了 54.58% 的正确率，略微高于使用 KNN 得到的最好结果 54.20%。

四、总结与反思

直接将 KNN 和支持向量机运用于基因位点预测分析并没有达到如期的效果，经过思考和讨论以后我们得出了几个可能的原因：

1. 样本数量过少。1000 个样本与 9445 个变量（基因位点编码信息）数量太少，即使是

SVM 这种具有强大泛化能力的方法仍不足以准确提炼出样本的特征。

2. 数据可靠性存在疑问，考虑到基因编码的私密性，准确真实的编码信息难以通过正常途径获取，我们获取的题目提供的数据也可能来自于数据模拟，本身与真实条件有所不同

参考资料：

- [1] 基因水平的关联分析方法 罗旭红, 刘志芳, 董长征
- [2] 精通机器学习：基于 R（第二版） Cory Leismester P.84
- [3] July.支持向量机通俗导论（理解 SVM 的三层境界）
http://blog.csdn.net/v_july_v/article/details/7624837.

附录

genotype.dat (27.1MB)	基因位点编码信息
phenotype.txt (1.95KB)	对应样本患病信息