贝叶斯课程论文——宝可梦数据分析

舒文炫 PB18000029

2021 年 12 月 31 日

摘要

宝可梦是一款全球范围内比较知名的以宠物养成，对战为核心玩法的游戏。二十余年来，该游戏仍然一直焕发她的活力，而本人也是这款游戏的老粉了。本人认为这归功于该公司优秀的宝可梦数值设计，使得每只宝可梦都有其用武之地，目前已经设计到第八世代，共计 800 余种宝可梦，每种都有其特色。本论文就是尝试使用贝叶斯方法去分析宝可梦数据，挖掘其内在的规律，一方面巩固所学贝叶斯知识，另一方面可以借此次机会深入理解那片神奇的宝可梦世界。

Key words: 宝可梦 贝叶斯分析 贝叶斯网络 主成分分析 贝叶斯回归

1 引言

1.1 研究问题描述

对于一只宝可梦其最基本的数值是各项种族值：生命，攻击，防御，特攻，特防，速度。种族值决定了该项能力的上限，种族值越高上限越高。生命防御和特防决定了该宝可梦的生存能力，攻击和特攻决定了其进攻能力，速度决定其是否在对战中有出手权。除了种族值还有宝可梦的属性，比如水火草雷冰之类。属性之间存在克制关系，最为直观的比如火克草，草克水，水克火。当用克制属性的技能达到对面的宝可梦时，伤害会直接翻倍，被克制的话就是一半。有的属性克制面很广但抗性不行，有的属性抗性面很广但输出不行，通过这些属性的搭配，可以组合出各种各样的战术。策略性上是很值得研究的。当然还有诸如性别，身高体重，捕获率这样的数据。对于每种宝可梦对应的还有

进化链，比如妙蛙种子 → 妙蛙草 → 妙蛙花。宝可梦培养到一定阶段会进化，进化越往后形态越高级，宝可梦也就越强。

那么我这里就想研究的是，这些数据之间是否存在关系，是线性关系，还是更复杂的关系？一只宝可梦的总体强度又是由什么来决定的？如果让我来设计宝可梦，我将如何去设计，使得数值分配合理？

1.2 数据集描述

本论文所使用的数据集来自 kaggle datasets 上面的 pokemon dataset，该数据集应是由 kaggle 用户自行整理贡献。这里面包含了第一世代到第七世代所有的宝可梦数据，共计数据项 721 条。每一条数据项的特征有 23 个，分别是序号，宝可梦名称，第一属性，第二属性，总种族值，六项种族值，所处世代，是否为神兽，主体颜色是什么，有无性别，雌雄比，第一蛋群，第二蛋群，能否超进化，身高，体重，捕获率，体型。

其中第二属性和第二蛋群都有缺失，因为不是所有的宝可梦都具有两个属性或者两个蛋群。雌雄比对于没有性别区分的宝可梦来说也是不存在的。其余数据都是完整的。

神兽是每个世代都会出的象征性的宝可梦，其往往拥有超高的种族值，且数目稀少，一个世代两到三只，他们在玩家之间的名气相较于其他宝可梦也更高，比如创世神兽阿尔宙斯。主体颜色就是这只宝可梦主要部分是什么颜色的，比如妙蛙种子是绿色的。蛋群涉及到宝可梦的繁衍，相同蛋群的宝可梦可以生蛋，不同蛋群宝可梦存在生殖隔离，也就是无法生蛋。捕获率就是在野外捕捉到这只宝可梦的概率，体型就是宝可梦的形状，比如妙蛙种子是 quadruped，也就是四足兽型，表示主要四条腿行走，更详细的介绍，有兴趣的读者可以自行查找宝可梦百科，这里只是让读者对数据的含义有一个基本认识，重点还是对数据的分析。

2 主要研究方法介绍

2.1 贝叶斯网络

贝叶斯网络又称信念网络，是一种概率图模型，于 1985 年由 Judea Pearl 首先提出，它是一种模拟人类推理过程中因果关系的不确定性处理模型，其网

络拓朴结构是一个有向无环图 (DAG)。

如果节点 A 直接影响节点 B，就用 $A \to B$ 来表示，其边上的权值即为条件概率 $P(B|A)$, 这里说的条件概率一般会是矩阵的形式，表示 A,B 两个变量里面的不同因子直接的关系. 朴素贝叶斯可以看成是贝叶斯网络的特殊情况，因为朴素贝叶斯假设所有节点是独立的。

关于贝叶斯网络的搭建，一方面可以使用专家知识搭建出一个网络，另一方面也可以使用算法去学习出结构，结构学习算法分为三类，基于约束，基于得分以及混合算法。基于约束算法来源于 Pearl 关于因果图模型的工作，其基本思想就是，给定节点 C，判断 A,B 是否条件独立，如果是，A,B 被 C 分割，不是则 A,B 直接相连。基于得分的算法，则是对备选的网络指定一个拟合优度评分，选取得分最高的网络。混合算法则是综合这些算法一起得到结果。学习得到结构之后，需要判断该结构是否真的能反应真实情况。然后利用得到的结构进行参数学习，得到每个节点之间的条件概率表，也称 CPT。

本论文将使用该方法对数据之间的条件依赖关系进行分析，尤其是对于因子数据，这种方法可以说是直观且有效。

2.2 主成分分析

[2] 一般的 PCA 基于特征值分解将高维数据线性投影到低维空间，这里我要使用的是 PPCA，也即 Probabilistic PCA。其假设存在低维空间上的隐变量 Z，且其先验服从正态分布 $N(0, I)$, 协方差阵为单位阵。而假设观测数据 X 来自于维数为 M 的空间

$$X = WZ + \mu + \epsilon$$

其中 W 是 $M \times D$ 维矩阵，μ 是 X 的均值，ϵ 是一个高斯误差，那么在给定 Z 的情况下，X 的条件分布为正态分布。那么 X 也就服从正态分布 $N(\mu, C)$。其中 $C = WW^T + \sigma^2 I$, 那么根据贝叶斯公式，我们能求出后验分布 $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$ 其也服从正态分布 $N(C^{-1}W^T(x-\mu), \sigma^{-2}M)$ 这就是我们所需要的结果。

该方法有两种实现，一是通过极大似然法，一是通过 EM 算法。对于极

大似然法，Tipping&Bishop 给出了一个闭式解

$$W_{ML} = U_M(L_M - \sigma^2 I)^{\frac{1}{2}} R \tag{2.1}$$

$$\sigma^2_{ML} = \frac{1}{D-M} \sum_{i=M+1}^{D} \lambda_i \tag{2.2}$$

其中 U_M 是 $D \times M$ 的矩阵，其列向量为协方差矩阵 S 的的任意 M 个特征向量。L_M 是对角阵，元素值为对应的特征值。R 是酉矩阵，理解为对投影矩阵进行旋转。通常情况下 R 直接取单位阵即可。Tipping&Bishop 证明了当 U_M 取最大的 M 个特征向量对应的特征值时，log likelihood 取最大。

由于是求极大似然估计，自然可以使用 EM 算法进行迭代，逐步更新参数 W 和 σ。

本论文将使用该方法对种族值，捕获率等连续型的数据进行主成分分析，对其进行降维，以期望可以归纳出对宝可梦强度等隐变量的认识。

2.3 贝叶斯 (广义) 线性回归

[1] 贝叶斯意义下的回归，我们假设响应变量服从分布 $Y \sim N(\beta X, \sigma^2 I)$，并通过给出 β 和 σ 的先验分布，最终得到这两个参数的后验分布，不过通常情况下，这个后验分布并不是一个简单的分布，要从中进行抽样需要使用 Gibbs 方法。

广义线性回归则是对线性回归的推广，不再使用简单的线性模型，而是通过非线性的联系函数将响应变量期望与线性预测量联系起来，用这种模型可以对更复杂的数据进行拟合回归，效果也会更好。

本论文将使用该方法对连续型数据进行回归分析，以期望得到这些数据之间较为直观的关系。

3 研究过程与结果

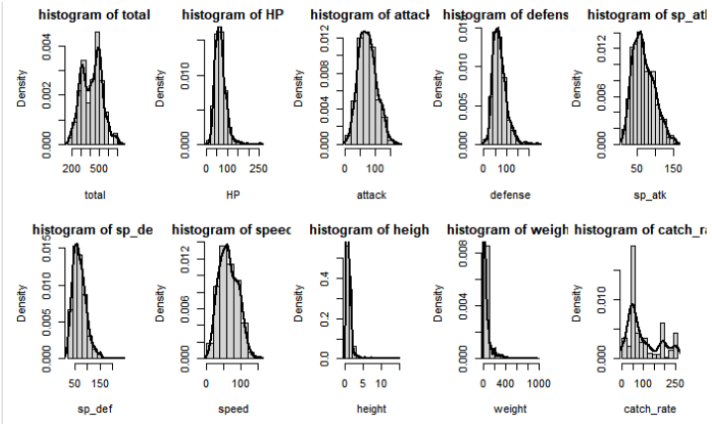
3.1 数据观察与处理

首先是对缺失值处理，这里缺失值只在第二属性，第二蛋群出现 (雌雄比在本论文中不予考虑) 我使用了比较简单的方法，就是如果第二属性和第二蛋

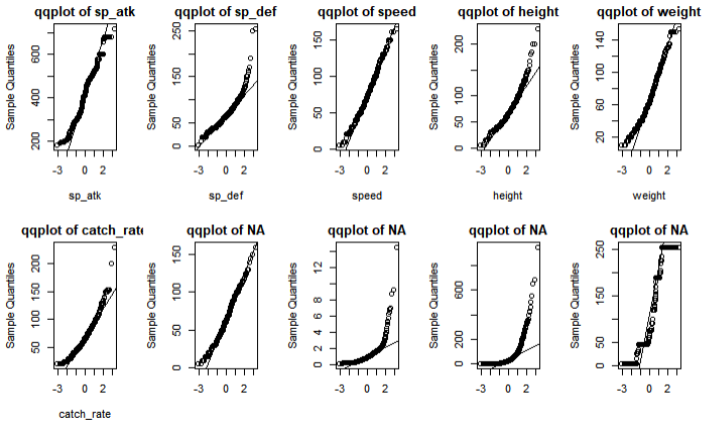
群的数据有缺失，那么直接使其分别等于第一属性和第一蛋群，因为这里的数据缺失并非是无法测量，而是该宝可梦只能具备一种属性或蛋群，那这样处理缺失值也是比较合理的做法。

对于异常值，这里我认为其不存在，毕竟所有的宝可梦设计出来，这个数据就是定下来的，数据的获得方面不会存在什么异常情况，其数据的分布即反应真实分布。

这里数据项只有 721 项，相对而言还只是一个小样本，这里唯一可能有的问题是数据的正态性问题，下面我对连续型的变量绘制直方图以查看其分布情况



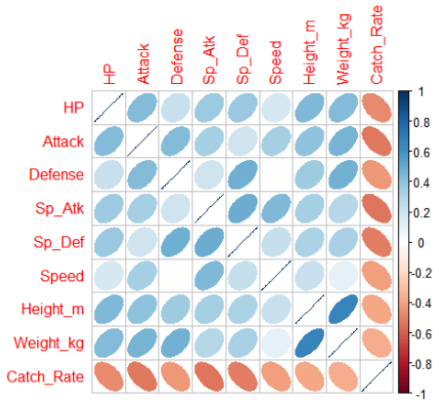
下面是每个变量对应的 qq 图



观察这些数据可以发现，实际上正态性并不是特别的好。第一个总种族值

看起来是一个双峰的分布，后面的捕获率也是，看起来是比较复杂的。这种情况对我后面的数据分析造成了一些麻烦。

下面再观察一下这些数据之间的线性相关性，利用 R 可以很方便的将其整合到一张图中，比较直观。

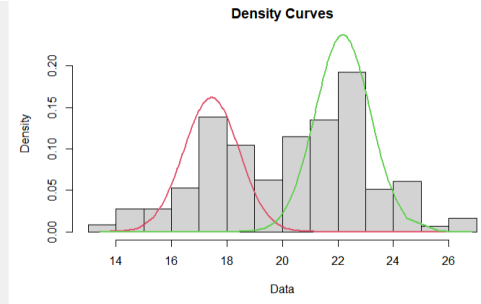


从这张图就可以观察发现，捕获率和其他的变量存在较好的线性相关性，身高和体重的线性相关性也比较的不错，这些都可以作为后面分析的一个方向。

4 结果展示与结论分析

4.1 分布拟合

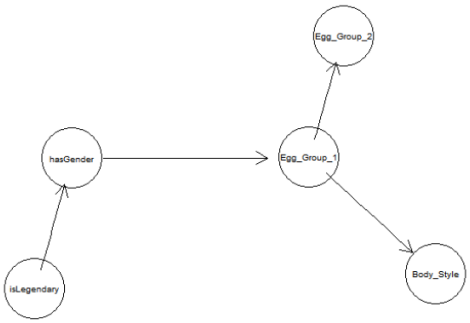
这部分我比较感兴趣的是种族值的具体分布，从前面画的图可以看出来这是一个双峰的分布，那么比较直观的感觉就是两个正态分布的混合，对于混合分布拟合比较好的方法是 EM 算法, 先假设数据来源于一个混合正态，$\lambda_1 N(\mu_1, \sigma_1^2) + \lambda_2 N(\mu_2, \sigma_2^2)$, 然后迭代最大化参数的后验分布，得到其后验众数估计。但是实际上只使用原数据拟合的情况不是很好，尝试了几种变换后，在选择开根号变换时得到了比较好的效果，拟合曲线如下



具体的分布 $\sqrt{Total} \sim 0.406N(17.4, 1) + 0.594N(22.2, 1)$. 相当于基本上宝可梦总种族值集中在 $17.4^2 = 302.76$ 和 $22.2^2 = 492.84$ 附近。前者种族值算比较低的，一般都是宝可梦的初级形态，后面是比较高的，一般是对应宝可梦的高级形态。可以看到这里的强度设计也是比较的均衡，相同形态的宝可梦基本差别不对，但是低级形态和高级形态的差别就比较明显，这样可以给人带来养成的满足感，同时也不会让人倾向于去培养某一只宝可梦，因为相同的形态的强度差不多，但是具有不同的功能，可以培养多只，组合在一起，配合作战，使得游戏在策略方面更具有可探索性。

4.2 贝叶斯网络方法

这部分是对因子变量的贝叶斯网络建模，剔除掉判断出相互独立的属性，我得到了如下的网络

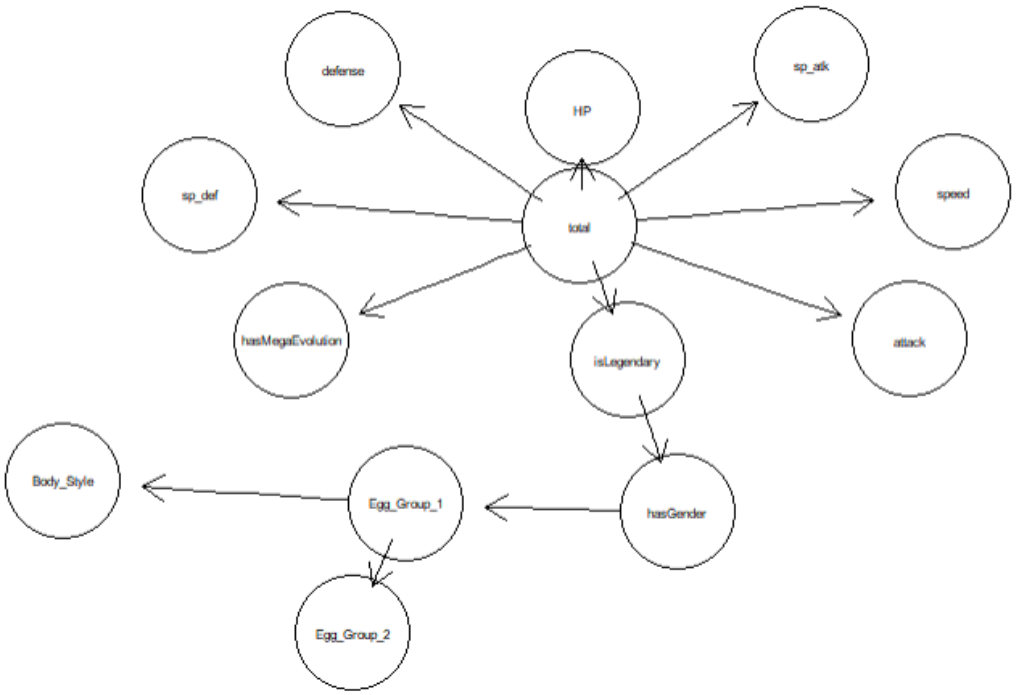


使用交叉验证方法对模型进行评估，得到 loss 是 4.97。

对该贝叶斯网络的解释：可以看到宝可梦是否具有性别是与其是否为神兽有关系的，这在宝可梦的设计中就体现出来了，神兽一般会设计成无性别

的。那么同理，如果没有性别自然也就无法繁殖，那么所处的蛋群自然也就不同。一般而言现实生活中，相同的种族会具有类似的体型，放到宝可梦中就是相同的蛋群对应的体型基本是一样的，这和我们的常识也是符合的，所以蛋群某种程度上也会决定宝可梦的体型。

贝叶斯网络如果想研究连续变量的关系，需要对连续变量进行离散化，这里我想更进一步的看一下加入种族值之后的网络表现，我选择的离散化方法是对每一项种族值进行聚类，类别设定为 5，基本就相当于按照种族值高低分成，低，较低，中等，较高，高，这 5 类。新得到的网络如下



使用交叉验证方法对模型进行评估，得到 loss 是 13.95。

这里得到的贝叶斯网络比较有意思，可以发现其主体是一个以 Total 为中心的辐射型的网络，在给定了 Total 的条件下，其他的节点相当于都是独立的。也就是说在相同的总种族值情况下，你没有办法去通过某一单项种族值来推断其他的种族值，不会存在什么某一单项种族值高，其他项的种族值就会高或者低。某种意义上，这样的分配可以使得宝可梦更为多样化，使得每种宝可梦都有自己的特色与用途，可以使得玩家不会觉得所有的宝可梦都是千篇一

律的。

同时可以看到总的种族值和能否超进化也就是 MegaEvolution 有关系。这是超进化的机制所决定，能进行超进化的宝可梦必须达到其最终形态，而宝可梦的最终形态的种族值相对于其初级形态是会高不少的。这就是可以通过宝可梦的种族值来推断其是否可以超进化。

4.3 主成分方法分析

下面是对种族值，身高，体重，捕获率这些特征进行主成分分析，使用 PPCA 方法允许我们不用对数据进行标准化，不过这里方便起见，我还是对数据进行了归一化。这里本身数据的维数并没有特别高，所以我直接使用前面提到的闭式解得到结果，不过我在代码中也尝试写了一下 EM 算法的实现，可以去附录代码中找到。

```
[1] 0.008522566
             [,1]         [,2]         [,3]
 [1,]  0.18694650  0.015020249 -0.045579494
 [2,]  0.05792685 -0.008785195 -0.015683740
 [3,]  0.12058515 -0.031053087 -0.076040774
 [4,]  0.06737388 -0.051015600 -0.037009855
 [5,]  0.14153418  0.084455421 -0.002207378
 [6,]  0.07981242  0.002368593 -0.003729686
 [7,]  0.09447361  0.091163823  0.006229552
 [8,]  0.03584012 -0.004672023 -0.016765297
 [9,]  0.04580375 -0.017900721 -0.030733135
[10,] -0.27050347  0.055027041 -0.085479316
```

第一行是 σ^2, 下面的矩阵每一行对应 x 投影到隐变量 Z 的空间时前面的系数。每一列就反应了每一个隐变量在 x 的每一个特征上面的作用。我这里隐变量取的是 3 维的。

重点看一下第一个主成分，观察到其种族值以及身高体重对应的系数都是正的，只有捕获率是负的，这一主成分可以解释为宝可梦隐含的强度属性。粗略来看肯定是种族值越高，宝可梦越强，而越强大的宝可梦在捕捉时会越困难。那么这些变量共同作用下，基本可以体现一只宝可梦的强度。同时因为我做了归一化，这些系数我们可以直接进行比较。

可以看到这里，绝对值最大的就是捕获率，其次是总种族值 total，这两项自不用说。然后是攻击 atk，特攻 sp_atk 以及速度 speed, 那么可以得出，前面两项越高越强是已经知道的，如果前面两项相同，更有决定性的就是宝可梦的攻击，特攻和速度。论文一开始介绍了速度决定了出手权，如果一只宝可梦速度快，且攻击或者特攻也高，那么显然其能先于对手出击，并且能保证击

杀对手，那自然就会更强。至于宝可梦的生存能力，也可以看到其防御和特防对应的系数是高于 HP 的，这说明防御和特防更为重要。

4.4 广义线性回归方法分析

这一部分我尝试分析捕获率和其种族值的关系，但是一般的线性模型并不能取得很好的效果，考虑到捕获率的分布也类似于一个双峰，我这里将捕获率分成了两类，一类低于 125，一类高于 125. 这样相当于对这个类别变量进行贝叶斯逻辑回归。最后结果比较不错, 有 84.33% 的准确率，其 AUC 值为 0.907, 说明该模型是不错的。

5 总结

本论文使用贝叶斯方法对宝可梦数据进行了较为全面的分析，也得出了一些比较有用的结论，实际上对于该数据集仍然能做很多工作，但这里限于篇幅无法继续说明。贝叶斯方法是一套十分有效的方法，通过本次分析，一方面巩固了贝叶斯分析的知识，另一方面也对宝可梦世界有了更深入的理解。

参考文献

[1] Hoff P. D. (2009) A first course in Bayesian Statistical methods. Springer.

[2] Christopher M. Bishop (2006) Pattern Recognition and Machine Learning. Springer.

6 附录

实验用到的 R 代码会附在这里

```
#读入宝可梦数据并对其进行基本的分析
pokemon<-read.csv("pokemon_alopez247.csv",
stringsAsFactors = TRUE)
str(pokemon)
summary(pokemon)
```

```
#选出连续型的数据绘制直方图查看分布
selindex1<-c(5,6,7,8,9,10,11,20,21,22)
seldata1<-pokemon[,selindex1]
feanames<-c("total","HP","attack","defense"
,"sp_atk","sp_def","speed","height"
,"weight","catch_rate")
opar<-par(no.readonly = TRUE)
par(mfrow=c(2,5))
for( i in c(1:10)){
hist(seldata1[,i],prob=TRUE,xlab=feanames[i]
,main=paste("histogram of",feanames[i]))
lines(density(seldata1[,i]),lwd=2)
}
par(opar)
```

```
#使用qq图查看分布正态性
opar<-par(no.readonly = TRUE)
par(mfrow=c(2,5))
for(i in selindex1){
qqnorm(pokemon[,i],xlab=feanames[i]
,main=paste("qqplot of",feanames[i]))
qqline(pokemon[,i])
```

```
    }
    par(opar)
```

```
    #绘制变量之间线性相关图
    library(corrplot)
    mat<-cor(pokemon[,selindex5])
    corrplot(mat,method="ellipse")
```

```
    #使用EM算法拟合混合正态
    library(mixtools)
    em <- normalmixEM(sqrt(pokemon[,5]),
    mu = c(0, 1), sigma = c(1, 1),
    sd.constr = c(1, 1))
    plot(em, whichplots = 2)
```

```
    #使用EM算法拟合混合正态
    library(mixtools)
    em <- normalmixEM(sqrt(pokemon[,5]),
    mu = c(0, 1), sigma = c(1, 1),
    sd.constr = c(1, 1))
    plot(em, whichplots = 2)
```

```
    #选取因子型数据学习贝叶斯网络结构
    library(bnlearn)
    library(Rgraphviz)
    selindex2<-c(3,4,13,14,15,17,18,19,23)
    seldata2<-pokemon[,selindex2]
    seldata2.bn<-hc(seldata2)
    graphviz.plot(seldata2.bn,layout="fdp")
    #将结构中有关系的拿出来，并对其进行参数学习
    selindex3<-c(13,15,17,18,23)
```

```
    seldata3<-pokemon[,selindex3]
    seldata3.bn<-hc(seldata3)
    graphviz.plot(seldata3.bn,layout="fdp")
    seldata3.fit<-bn.fit(seldata3.bn,
    data=seldata3)
    print(seldata3.fit)
    #对得到的模型使用交叉验证方法进行检验
    modelx<-bn.cv(seldata3 ,seldata3.bn,k=10)
    plot(modelx)
    loss(modelx)
```

```
#对连续型数据进行kmeans聚类，
#得到处理后的因子型变量
library(infotheo)
library(tidyverse)
library(dplyr)

##a3<-discretize(pokemon[,7],"equalwidth",5)
a1<-kmeans(pokemon[,5],5)$cluster
a2<-kmeans(pokemon[,6],5)$cluster
a3<-kmeans(pokemon[,7],5)$cluster
a4<-kmeans(pokemon[,8],5)$cluster
a5<-kmeans(pokemon[,9],5)$cluster
a6<-kmeans(pokemon[,10],5)$cluster
a7<-kmeans(pokemon[,11],5)$cluster

a<-cbind(a1,a2,a3,a4,a5,a6,a7)

b1<-as.character(a[,1])
b2<-as.character(a[,2])
b3<-as.character(a[,3])
b4<-as.character(a[,4])
```

```
b5<-as.character(a[,5])
b6<-as.character(a[,6])
b7<-as.character(a[,7])
b<-cbind(b1,b2,b3,b4,b5,b6,b7)
b<-data.frame(b,stringsAsFactors = TRUE)
names(b)<-c("total","HP","attack",
"defense","sp_atk",
"sp_def","speed")
procdata<-cbind(b,pokemon[,c(13,15,17,18,19,23)])
str(procdata)
summary(procdata)
#对处理好的数据进行贝叶斯网络的搭建
procdata.bn<-hc(procdata)
graphviz.plot(procdata.bn,layout="fdp")
procdata.fit=bn.fit(procdata.bn,procdata)
#对模型进行评估
modelx1<-bn.cv(procdata,procdata.bn,k=10)
plot(modelx1)
loss(modelx1)
```

```
  #EM算法实现PPCA
  ppca<-function(mu,predimension
  ,posdimension,rawdata,steps){
  set.seed(2)
  sigma<-1
  W<-matrix(rnorm(predimension*posdimension)
  ,nrow=predimension)
  M<-t(W)%*%W+diag(posdimension)*sigma
  invM<-solve(M)
  covW<-t(W)%*%W
##初始值

```

```
13
14    sumEz<-matrix(rep(0,
15    times=posdimension*predimension),
16    nrow=predimension)
17    sumEzz<-matrix(rep(0,
18    times=posdimension*posdimension),
19    nrow=posdimension)
20  for(k in c(1:steps)){
21
22    for(i in c(1:721)){
23      Ez<-invM%*%t(W)
24      %*%matrix(t(rawdata[i,])-mu,ncol=1)
25      Ezz<-sigma*invM+Ez%*%t(Ez)
26      sumEz<-sumEz+matrix(
27            t(rawdata[i,])-mu,ncol=1)%*%t(Ez)
28      sumEzz<-sumEzz+Ezz
29
30    }
31    newW<-sumEz%*%solve(sumEzz)
32    covnewW<-t(newW)%*%newW
33    sumsigma<-0
34    for(i in c(1:721)){
35      sumsigma<-sumsigma+
36      t(matrix(t(rawdata[i,])-mu,ncol=1))
37      %*%matrix(t(rawdata[i,])
38      -mu,ncol=1)
39      -2*t(invM%*%t(W)%*%
40      matrix(t(rawdata[i,])-mu,ncol=1))
41      %*%t(newW)%*%matrix(t(rawdata[i,])-mu,ncol=1)
42      +sum(diag((sigma*invM+Ez%*%t(Ez))%*%covnewW))
43    }
44    if(abs(sigma
```

```
  -sumsigma[1,1]/721/predimension)<0.000001){
    break;
  }
  W<-newW
  sigma<-sumsigma[1,1]/721/predimension
  M<-t(W)%*%W+sigma*diag(posdimension)
  invM<-solve(M)
  covW<-t(W)%*%W
}
print(sigma)
return(W)
}
```

```
    #直接使用闭式解求PPCA
    selindex4<-c(5:11,20:22)
    seldata4<-pokemon[,selindex4]
    useseldata<-matrix(rep(0,times=7210),nrow=721)
for(i in c(1:10)){
  for(j in c(1:721)){
    useseldata[j,i]=(max(seldata4[,i])
    -seldata4[j,i])/(max(seldata4[,i])
    -min(seldata4[,i]))
  }

}
datacor<-cor(seldata4)
mu<-colMeans(useseldata)
posdimension<-3
print(ppca(mu,10,posdimension,useseldata,1000))

S<-cov(useseldata)
P<-eigen(S)
```

```
print(P)
U<-P$vectors[,1:3]
sigma<-mean(P$values[4:10])
print(sigma)
L<-diag(3)
L[1,1]<-sqrt(P$values[1]-sigma)
L[2,2]<-sqrt(P$values[2]-sigma)
L[3,3]<-sqrt(P$values[3]-sigma)
W1<-U%*%L
print(W1)
```

```
#对捕获率进行二分类
library(plyr)
predata<-mutate(pokemon[,selindex5],
      Catch_Rate=ifelse(Catch_Rate>125,0,1)
           )
str(predata)
sum(predata[,9])/721
#进行logistic回归
fit3<-brm(Catch_Rate~HP+Attack+
Defense+Sp_Atk+Sp_Def+Speed,
data=predata,chains=2,
cores=2,iter=5000,
family = bernoulli)
library(ggplot2)
summary(fit3)
stanplot(fit3,
     type = "trace")
#模型准确率
library(dplyr)
Pred <- predict(fit3, type = "response")
Pred <- ifelse(Pred[,1] > 0.5, 1, 0)
```

```
ConfusionMatrix <- table(Pred,
pull(predata, Catch_Rate))
#correct classification rate
sum(diag(ConfusionMatrix))/sum(ConfusionMatrix)
#模型分类效果评估
library(ROCR)
# Compute AUC for predicting Class with the model
Prob <- predict(fit3, type="response")
Prob <- Prob[,1]
Pred <- prediction(Prob,
as.vector(pull(predata, Catch_Rate)))
AUC <- performance(Pred, measure = "auc")
AUC <- AUC@y.values[[1]]
AUC
```