# Machine Learning 2019

袁欣

2019 年 5 月 28 日

# 1 朴素贝叶斯与 EM 算法

## 1.1 朴素贝叶斯实验数据

读入朴素贝叶斯实验数据，并进行展示。

```
da <- read.csv("data.csv")
da.train <- da[1:14, ]
da.test <- da[15, ]
```

- 数据展示

| No | Age | Income | Student | Credit_rating | Class.buys_computer |
|---|---|---|---|---|---|
| 1 | <=30 | High | No | Fair | No |
| 2 | <=30 | High | No | Excellent | No |
| 3 | 31-40 | High | No | Fair | Yes |
| 4 | >40 | Medium | No | Fair | Yes |
| 5 | >40 | Low | Yes | Fair | Yes |
| 6 | >40 | Low | Yes | Excellent | No |
| 7 | 31-40 | Low | Yes | Excellent | Yes |
| 8 | <=30 | Medium | No | Fair | No |
| 9 | <=30 | Low | Yes | Fair | Yes |
| 10 | >40 | Medium | Yes | Fair | Yes |
| 11 | <=30 | Medium | Yes | Excellent | Yes |
| 12 | 31-40 | Medium | No | Excellent | Yes |
| 13 | 31-40 | High | Yes | Fair | Yes |
| 14 | >40 | Medium | No | Excellent | No |
| 15 | <=30 | Medium | Yes | Fair | |

## 1.2 朴素贝叶斯简介

朴素贝叶斯分类器（naive Bayes classifier）采用了 “属性条件独立性假设”：对已知类别，假设所有属性相互独立。换而言之，假设每个属性独立地对分裂结果发生影响。

- 代码实现

```
naiveBayes <- function(da, Classn = 6, Factorn = c(2:5)){
  #
  Class <- levels(as.factor(da[, Classn]))
  Pc <- vector(length = length(Class))
  for(i in 1:length(Class)){
    Pc[i] <- (length(da[which(da[, Classn] == Class[i]), Classn]) + 1)/(nrow(da) + 2)
  }
  Pc <- data.frame(Class = Class, P = Pc)
  #
  Factorlist <- list()
  k <- 1
  for(i in Factorn){
    Classi <- levels(as.factor(da[, i]))
    temp <- as.data.frame(matrix(nrow = length(Classi), ncol = nrow(Pc) + 1))
    temp[, 1] <- Classi
    for(j in 1:length(Classi)){
      for(t in 1:length(Class)){
        temp[j, t + 1] <- (length(da[which(da[, i] == Classi[j] & da[, Classn] == Class[t]), i]) + 1)
      }
    }
    colnames(temp) <- c("Class", Pc$Class)
    Factorlist[[k]] <- temp
    k <- k+1
  }
  names(Factorlist) <- colnames(da[, Factorn])
  return(list(Pc = Pc, Factorlist = Factorlist))

}

mymodel <- naiveBayes(da.train, 6, c(2:5))

predictnB <- function(da, mymodel, Factorn = c(2:5)){
  #
  factors <- colnames(da[, Factorn])
```

```r
  Classn <- nrow(mymodel$Pc)
  P <- data.frame(Classn = mymodel$Pc$Class, P = 1)
  for(i in 1:Classn){
    for(j in 1:length(factors)){
      temp <- mymodel$Factorlist[factors[j]][[1]]
      ind <- which(temp$Class == da[,factors[j]])
      P$P <- as.numeric(P$P * temp[ind, -1])
    }

  }

  return(P)
}

knitr::kable(predictnB(da.test, mymodel, c(2:5)))
```

| Classn | P |
|---|---|
| No | 0.0004036 |
| Yes | 0.0014952 |

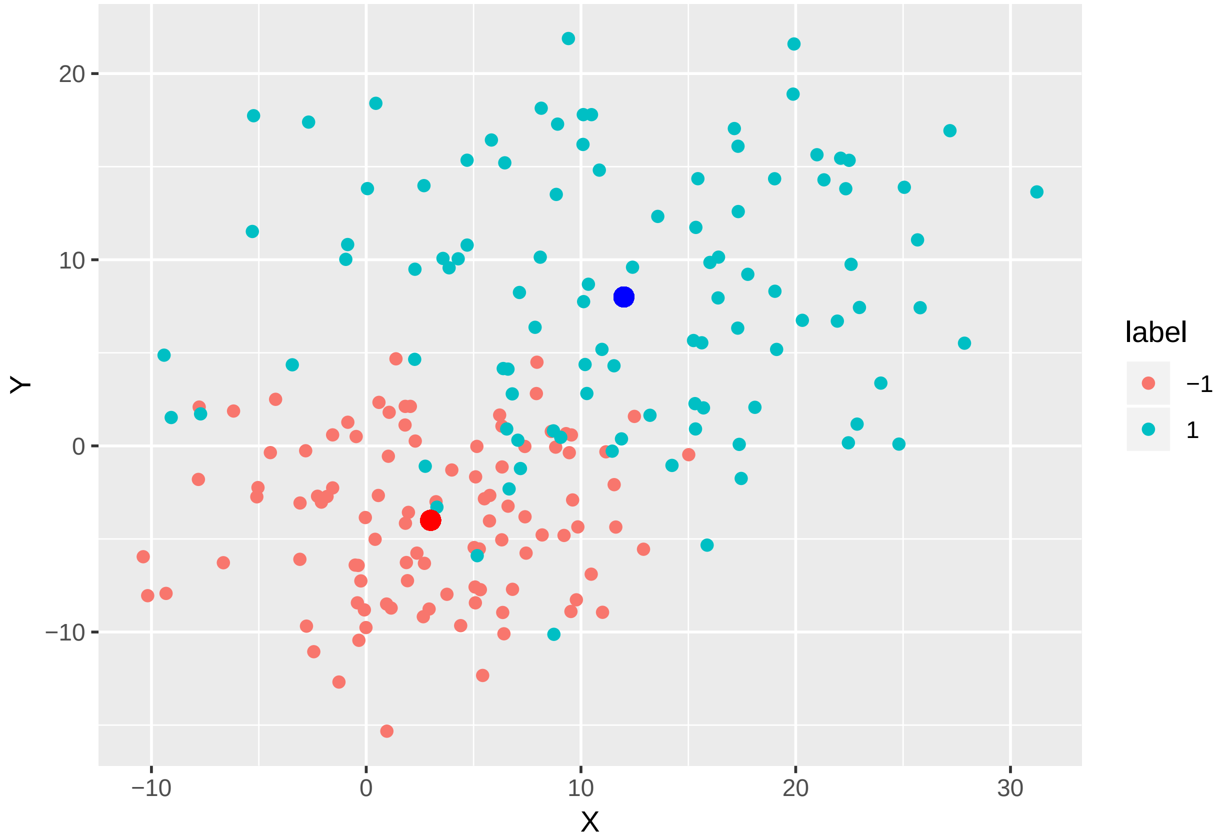
由上表可以看出 Yes 的概率大于 No 的概率，所以测试样本应该会买电脑。

## 1.3 EM 算法实验数据

```r
CreatData <- function(n, mu1, mu2, Sigma1, Sigma2, seed = 3){
  set.seed(seed)
  X1 <- mvrnorm(n, mu1, Sigma1)
  set.seed(seed)
  X2 <- mvrnorm(n, mu2, Sigma2)
  df <- data.frame(X = c(X1[, 1], X2[, 1]), Y = c(X1[, 2], X2[, 2]),
                   label = factor(c(rep(-1, n), rep(1, n))))
  return(df)
}
da <- CreatData(100, c(3, -4), c(12, 8), 25*diag(2), 64*diag(2))

ggplot(data = da, aes(x = X, y = Y, colour = label)) +
  geom_point(size = 2.0, shape = 16) +
  geom_point(aes(x = 3, y = -4), color = "red", size = 3) +
```

```
geom_point(aes(x = 12, y = 8), color = "blue", size = 3)
```




## 1.4 EM 算法估计高斯混合分布参数