mml87_nb

Kelvin

27 June 2017

MLE of NB parameters

The MLE estimations of NB parameters are as follows:

$$P(Y = y) = \frac{count(y)}{n}$$
$$P(X_j = x|Y = y) = \frac{count(x) \cup count(y)}{count(y)}$$

Notice that the log likelihood function that stated below is different from what mostly shown by others, which do not consider the case where $Y_i = T$ and $Y_i = F$. Hence, it is not sure whether we can use the above MLE of parameters. The following is the implementations of MLE of NB parameters.

```
# y is the output node
# x is a set of input nodes
# estimated parameters are stored in a list
# smoothing is additive smoothing constant to avoid 0 probabilities but it can take any other values
# y parameters is always store at the end
mle_est_nb = function(y, x, data, smoothing = 1) {
  xIndices = which(colnames(data) %in% x)
  yIndex = which(colnames(data) == y)
  lst = list()
  for (i in 1:length(x)) {
    lst[[i]] = t((table(data[, c(yIndex, xIndices[i])]) + smoothing) /
      (rowSums(table(data[, c(yIndex, xIndices[i])])) + smoothing * nlevels(data[, xIndices[i]])))
  } # end for i
  # always store y parameters at the last
  lst[[length(x) + 1]] = (table(data[, yIndex]) + smoothing) / (nrow(data) + smoothing * nlevels(data[,
  names(lst) = c(x, y)
  return(lst)
}
```

Negative log likelihood of NB

For a Naive Bayes model with binary variable, its parameters are $\{P(Y_i = T), P(X_{ij}|Y_i = T), P(X_{ij}|Y_i = F)\}$. To simply the notations, we use $\{p_{i0}, p_{ij1}, p_{ij2}\}$ to denote the above probabilities respectively. The likelihood of Naive Bayes given a data set is then

$$l = \prod_{i}^{n} P(Y_i = T|\vec{X}_i)^{Y_i}(1 - P(Y_i = T|\vec{X}_i))^{1-Y_i}$$

where the posterior probability of Y_i given a vector $\vec{X_i} =< X_{i1}, \ldots, X_{im} >$ is

$$\begin{aligned}
P(Y_i = T|\vec{X_i}) &= \frac{P(Y_i = T) \prod_{j=1}^m P(X_{ij}|Y_i = T)}{P(\vec{X_i})} \\
&= \frac{P(Y_i = T) \prod_{j=1}^m P(X_{ij}|Y_i = T)}{P(Y_i = T) \prod_{j=1}^m P(X_{ij}|Y_i = T) + (1 - P(Y_i = T)) \prod_{j=1}^m P(X_{ij}|Y_i = F)} \\
&= \frac{p_{i0} \prod_{j=1}^m p_{ij1}}{p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2}}
\end{aligned}$$

The negative loglikelihood

$$\begin{aligned}
L &= -\sum_{i=1}^n \left[Y_i \ln p(Y_i|\vec{X_i}) + (1 - Y_i) \ln(1 - p(Y_i|\vec{X_i})) \right] \\
&= -\sum_{i=1}^n \left[Y_i \ln p_{i0} + (1 - Y_i) \ln(1 - p_{i0}) + Y_i \sum_{j=1}^m \ln p_{ij1} + (1 - Y_i) \sum_{j=1}^m \ln p_{ij2} - \ln \left(p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2} \right) \right]
\end{aligned}$$

NLL implementations

```
nll_auxiliary = function(dataPoint, pars, xIndices, yIndex) {
  ss = 0
  for (j in 1:(length(pars) - 1)) {
    ss = ss + log(p_ijk(dataPoint, pars, xIndices, xIndices[j], dataPoint[[yIndex]]))
  }
  ss = ss + log(pars[[length(pars)]][dataPoint[[yIndex]]])
  return(ss)
}
```

Fisher information matrix

The first derivatives of the above negative log likelihood w.r.t. each parameter are

$$\begin{aligned}
\frac{\partial L}{\partial p_{i0}} &= -\sum_{i=1}^n \left[\frac{Y_i}{p_{i0}} - \frac{1 - Y_i}{1 - p_{i0}} - \frac{\prod_{j=1}^m p_{ij1} - \prod_{j=1}^m p_{ij2}}{p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2}} \right] \\
\frac{\partial L}{\partial p_{ik1}} &= -\sum_{i=1}^n \left[\frac{Y_i}{p_{ik1}} - \frac{p_{i0} \prod_{j=1}^m p_{ik1}}{p_{ik1} \left(p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2} \right)} \right] \\
\frac{\partial L}{\partial p_{ik2}} &= -\sum_{i=1}^n \left[\frac{1 - Y_i}{p_{ik2}} - \frac{(1 - p_{i0}) \prod_{j=1}^m p_{ik2}}{p_{ik2} \left(p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2} \right)} \right]
\end{aligned}$$

The second derivatives are

$$
\begin{aligned}
\frac{\partial^2 L}{\partial p_{i0}^2} &= \sum_{i=1}^{n} \left[\frac{Y_i}{p_{i0}^2} + \frac{1-Y_i}{(1-p_{i0})^2} - \left(\frac{\prod_{j=1}^m p_{ij1} - \prod_{j=1}^m p_{ij2}}{p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2}} \right)^2 \right] \\
\frac{\partial^2 L}{\partial p_{ik1}} &= \sum_{i=1}^{n} \left[\frac{Y_i}{p_{ik1}^2} - \left(\frac{p_{i0} \prod_{j=1}^m p_{ij1}}{p_{ik1} \left(p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2} \right)} \right)^2 \right] \\
\frac{\partial^2 L}{\partial p_{ik2}} &= \sum_{i=1}^{n} \left[\frac{1-Y_i}{p_{ik2}^2} - \left(\frac{(1-p_{i0}) \prod_{j=1}^m p_{ij2}}{p_{ik2} \left(p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2} \right)} \right)^2 \right] \\
\frac{\partial^2 L}{\partial p_{i0} p_{ik1}} &= \sum_{i=1}^{n} \frac{\prod_{j=1}^m p_{ij1} p_{ij2}}{\left(p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2} \right)^2} \frac{1}{p_{ik1}} \\
\frac{\partial^2 L}{\partial p_{i0} p_{ik2}} &= \sum_{i=1}^{n} \frac{\prod_{j=1}^m p_{ij1} p_{ij2}}{\left(p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2} \right)^2} \frac{-1}{p_{ik2}} \\
\frac{\partial^2 L}{\partial p_{ik1} p_{ik2}} &= \sum_{i=1}^{n} \frac{\prod_{j=1}^m p_{ij1} p_{ij2}}{\left(p_{i0} \prod_{j=1}^m p_{ij1} + (1-p_{i0}) \prod_{j=1}^m p_{ij2} \right)^2} \frac{-p_{i0}(1-p_{i0})}{p_{ik1} p_{ik2}}
\end{aligned}
$$

Since FIM entries are expectations of the second derivatives, we need to take expectations for the first three second derivatives that contain Y_i. To simplify the notations, we use p_x to denote $p_{i0} \prod_{j=1}^m p_{ij1} + (1 - p_{i0}) \prod_{j=1}^m p_{ij2}$. Then the expectations become

$$
\begin{aligned}
E\left(\frac{\partial^2 L}{\partial p_{i0}^2} \right) &= \sum_{i=1}^{n} \left[\frac{\prod_{j=1}^m p_{ij1}}{p_{i0} p_x} + \frac{\prod_{j=1}^m p_{ij2}}{(1-p_{i0}) p_x} - \left(\frac{\prod_{j=1}^m p_{ij1} - \prod_{j=1}^m p_{ij2}}{p_x} \right)^2 \right] \\
E\left(\frac{\partial^2 L}{\partial p_{ik1}} \right) &= \sum_{i=1}^{n} \left[\frac{p_{i0} \prod_{j=1}^m p_{ij1}}{p_{ik1}^2 p_x} - \left(\frac{p_{i0} \prod_{j=1}^m p_{ij1}}{p_{ik1} p_x} \right)^2 \right] \\
E\left(\frac{\partial^2 L}{\partial p_{ik2}} \right) &= \sum_{i=1}^{n} \left[\frac{(1-p_{i0}) \prod_{j=1}^m p_{ij2}}{p_{ik2}^2 p_x} - \left(\frac{(1-p_{i0}) \prod_{j=1}^m p_{ij2}}{p_{ik2} p_x} \right)^2 \right]
\end{aligned}
$$

FIM implementations

```r
# p(x_ij|y=k)
# xIndices is a vector of x indices in data
# xIndex is the index of the particular x that we want
p_ijk = function(dataPoint, pars, xIndices, xIndex, yValue) {
  xValue = dataPoint[[xIndex]]
  xParsIndex = which(xIndices == xIndex)
  return(pars[[xParsIndex]][xValue, yValue])
}

# \prod_j p(x_ij|y=k)
# i is the dataPoint index
# j is the X index in data
# k is the value of Y, i.e. k \in [1, arity(y)]
```

```
prod_pijk = function(dataPoint, pars, xIndices, yValue) {
  mm = 1
  for (i in 1:length(xIndices)) {
    xValue = dataPoint[[xIndices[i]]]
    mm = mm * pars[[i]][xValue, yValue]
  }
  return(mm)
}
```

```
dag = empty.graph(c("X1", "X2", "Y", "X3"))
dag = set.arc(dag, "Y", "X1")
dag = set.arc(dag, "Y", "X2")
dag = set.arc(dag,  "Y", "X3")
#set.seed(10086)
cpts = randCPTs(dag, 2, 1)
data = rbn(cpts, 10)
nVars = ncol(data)

#data = data[, sample(1:ncol(data))]
y = "Y"
x = c("X1", "X2", "X3")
arities = sapply(data, nlevels)
yIndex = which(colnames(data) == y)
xIndices = which(colnames(data) %in% x)

# mle of parameters with smoothing
pars = mle_est_nb(y, x, data, 1)

# p(y=T)
py1 = pars[[length(pars)]][[1]]
# p(y=F)
py2 = pars[[length(pars)]][[2]]

# a vector of \prod_j p(x_ij|y=1)
prodPij1 = apply(data, 1, prod_pijk, pars = pars, xIndices = xIndices, yValue = 1)
# a vector of \prod_j p(x_ij|y=2)
prodPij2 = apply(data, 1, prod_pijk, pars = pars, xIndices = xIndices, yValue = 2)

# a vector of p_xi
px = py1 * prodPij1 + py2 * prodPij2

# a matrix of p(x_j|y=T) and p(y_j|y=F)
probsMatrix = c()
for (j in xIndices) {
  probsMatrix = cbind(probsMatrix, apply(data, 1, p_ijk, pars = pars, xIndices = xIndices, xIndex = j, 
}
```

Off diagonal of FIM

```
# empty FIM
fimDim = (arities[yIndex] - 1) + length(x) * arities[yIndex]
fim = matrix(0, nrow = fimDim, ncol = fimDim)
```

```
# off diagonal entries
mm = prodPij1 * prodPij2 / (px ^ 2) # a common contant
fim[1, -1] = colSums(mm / probsMatrix) # fill in 1st row of FIM
fim[1, odd(2:ncol(fim))] = -1 * fim[1, odd(2:ncol(fim))]
for (rowIndex in 2:(nrow(fim) - 1)) {
  if (ncol(fim) - rowIndex == 1) {
    fim[rowIndex, -(1:rowIndex)] = sum(-mm * py1 * py2 / (probsMatrix[, rowIndex - 1] * probsMatrix[, -
  } else {
    fim[rowIndex, -(1:rowIndex)] = colSums(-mm * py1 * py2 / (probsMatrix[, rowIndex - 1] * probsMatrix
  } # end else
}

fim = fim + t(fim) # duplicate upper to lower triangular fim
#fim
```

Diagonal of FIM

```
# assume all variables are binary, hence the diag[1] is always the 2nd derivative w.r.t. p_i0
diag(fim)[1] = sum(prodPij1 / (py1 * px) + prodPij2 / (py2 * px) - ((prodPij1 - prodPij2) / px) ^ 2)

# two vectors to store 2nd derivative w.r.t. p_ik1 and p_ik2 respectively, where the 1 and 2 correspond
# z = c()
# for (i in 1:length(xIndices)) {
#   v = sum(py1 * py2 * prodPij1 * prodPij2 / (px * probsMatrix[, i * 2 - 1]) ^ 2)
#   w = sum(py1 * py2 * prodPij1 * prodPij2 / (px * probsMatrix[, i * 2]) ^ 2)
#   z = c(z, v, w)
# }
#

diag(fim)[-1] = colSums(py1 * py2 * prodPij1 * prodPij2 / (px * probsMatrix) ^ 2)
#fim
#det(fim) # negative values appear again!
#log(det(fim))
```

MML of Naive Bayes

$$I = -\ln K - \ln h(\vec{\theta}) + \frac{1}{2}\ln F(\vec{\theta}) - \ln f(D|\vec{\theta}) + \frac{|\vec{\theta}|}{2}$$

where $\vec{\theta} =< p_{i0}, p_{ij1}, p_{ij2} >, \forall j \in [1, m]$ is the set of parameters, $|\vec{\theta}|$ is the number of free parameters, K is the lattice contant and $h(\vec{\theta})$ is the parameter prior. A commonly used conjugate prior for binary variables is beta prior (i.e., beta distribution) with probability density function

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

where $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$. For simplicity, we assume all parameters are uniformly (i.e., $\alpha = \beta = 1$), hence for a single parameter prior is $x(1-x)$. Assuming all parameters are independent, we have

$$\ln h(\vec{\theta}) = \ln p_{i0} + \ln(1 - p_{i0}) + \sum_{j=1}^{m} [\ln p_{ij1} + \ln(1 - p_{ij1}) + \ln p_{ij2} + \ln(1 - p_{ij2})]$$

Substituting this into the above MML formula we get

$$I = -\left(\ln p_{i0} + \ln(1-p_{i0}) + \sum_{j=1}^{m}\left[\ln p_{ij1} + \ln(1-p_{ij1}) + \ln p_{ij2} + \ln(1-p_{ij2})\right]\right) + \frac{1}{2}F(\vec{\theta}) - \ln f(D|\vec{\theta}) + \frac{d}{2}(1+\ln k_d)$$

where d is the number of free parameters and k_d is the lattice constant for each free parameter.

```
nll = -sum(apply(data, 1, nll_auxiliary, pars = pars, xIndices = xIndices, yIndex = yIndex)) + sum(log(p
nll
```

```
## [1] 2.802587
```

```
f = det(fim)
log(f)
```

```
## [1] 20.72456
```

```
# test
```