mml87_nb

Kelvin

27 June 2017

```
dag = empty.graph(c("X1", "X2", "Y"))
dag = set.arc(dag, "Y", "X1")
dag = set.arc(dag, "Y", "X2")
cpts = randCPTs(dag, 2, 1)
data = rbn(cpts, 1000)
# model = naiveBayes(Y ~ ., data = data)
# model$tables
```

Log likelihood

For NB with multinomial variables, there are two types of parameters $P(Y = y)$ and $P(X_j = x|Y = y)$. The likelihood of Naive Bayes is then

$$\begin{aligned} P(Y|\vec{X}) &= \frac{P(Y)\prod_{j=1}^{m} P(X_j|Y)}{P(\vec{X})} \\ &= \frac{P(Y)\prod_{j=1}^{m} P(X_j|Y)}{\sum_Y P(Y)\prod_{j=1}^{m} P(X_j|Y)} \end{aligned}$$

where $\vec{X} =< X_1, \ldots, X_m >$ is a vector of m input variables. The negative loglikelihood of an entire data set with n observations is

$$\begin{aligned} L &= -\sum_{i=1}^{n} \log P(Y|X) \\ &= -\sum_{i=1}^{n} \left[\log P(Y) + \sum_{j=1}^{m} \log P(X_j|Y) - \log \left(\sum_Y P(Y) \prod_{j=1}^{m} P(X_j|Y) \right) \right] \end{aligned}$$

Since it is not known whether or not there is a closed form solution for the determinant of the FIM, we firstly use MLE of NB parameters

$$P(Y = y) = \frac{count(y)}{n}$$

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

The following are implementations of MLE of NB parameters.

MLE implementations

A function to calculate maximum likelihood estimation 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
# it can take any other values
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]] = (table(data[, c(yIndex, xIndices[i])]) + smoothing) /
      (rowSums(table(data[, c(yIndex, xIndices[i])])) + smoothing * nlevels(data[, xIndices[i]]))
  } # end for i
  lst[[yIndex]] = (table(data[, yIndex]) + smoothing) / (nrow(data) + smoothing * nlevels(data[, 3]))
  names(lst) = c(x, y)
  return(lst)
}

# (mle_est_nb("Y", c("X1", "X2"), data, 0)) # with no smoothing
(pars = mle_est_nb("Y", c("X1", "X2"), data, 1)) # with smoothing
```

Fisher information matrix

Define $\theta_0 = P(Y = y)$ and $\theta_j = P(X_j = x|Y = y), \forall j \in [1, m]$. The second derivative of the above negative log likelihood w.r.t each parameter is

$$\frac{\partial^2 L}{\partial \theta_k^2} = \sum_{i=1}^{n} \left[\frac{1}{\theta_k^2} - \left(\frac{\prod_{j=0}^{\neq k} \theta_j}{\sum_y \prod_{j=0}^{m} \theta_j} \right)^2 \right]$$

$$\frac{\partial^2 L}{\partial \theta_k \partial \theta_l} = \sum_{i=1}^{n} \left[\frac{(\sum_y \prod_{j=0}^{m} \theta_j)(\sum_{j=0}^{\neq k,l} \theta_j) - (\prod_{j=0}^{\neq k} \theta_j)(\prod_{j=0}^{\neq l} \theta_j)}{\left(\sum_y \prod_{j=0}^{m} \theta_j\right)^2} \right]$$

FIM implementations

```
px = function(y, x, data, dataPoint) {
  arities = sapply(data, nlevels)
  yIndex = which(colnames(data) == y)
  xIndices = which(colnames(data) %in% x)
  ss = 0
  for (yValue in 1:arities[yIndex]) {
    mm = pars[[yIndex]][yValue]
    for (j in 1:length(x)) {
      xValue = data[dataPoint, xIndices[j]]
      mm = mm * pars[[j]][yValue, xValue]
    } # end for each x_j
    ss = ss + mm
  } # end for each y value
  return(ss)
}

head(data)
pars
px("Y", c("X1", "X2"), data, 1)
```