Multiclass classification example with Logistic Regression

06/11/2020

Snaider-70- Enrique A. ELVER---

Executive Summary

Problem

The logistic regression algorithm is commonly used for classification problems. In this paper, I tried by classify iris species with logistic regression.

Results

The result shows that the accuracy of classifying setosa and non-setosa is 100% and classifying virginica and non-virginica is above 95%. The remaining data is versicolor and those are classified with 100% accuracy.

Conclusion

The multiclass logistic regression model performs well for classifying iris species and in this specific example, significant features are different between Bayesian and frequentist approach.

Introduction

In this paper, I tried to classify the species of IRIS data set with a logistic regression algorithm. To implement multiclass classification, I created two dummy variables by one-vs.-rest strategy and I discovered parameters by rjags library.

Data

IRIS data is one of the most famous dataset among R users. Please refer the Iris flower data set

Model and Results

Scatter plot between Petal.Width and Petal.Length by species suggests that setosa is distinguishable. Versicolor and virginica are adjacent but still separable.

I created two dummy variables as below

```
iris$class1_dummy = ifelse(iris$Species == "setosa", 1, 0)
iris$class2_dummy = ifelse(iris$Species == "virginica", 1, 0)
```

To apply logistic regression, I normalized predictive variables.

```
iris_normalized = iris
X = scale(iris[1:4], center=TRUE, scale=TRUE)
X = cbind(X, class1_dummy = iris$class1_dummy, class2_dummy = iris$class2_dummy)
```

Classifying setosa and non-setosa

I followed the lesson 9 example code as below.

```
library("rjags")
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
set.seed(725)

mod1_string = " model {
  for(i in 1:length(y)) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = int + b[1] * Sepal.Length[i] + b[2] * Sepal.Width[i]
                      + b[3] * Petal.Length[i] + b[4] * Petal.Width[i]
  }

  int ~ dnorm(0.0, 1.0/25.0)
  for(j in 1:4) {
    b[j] ~ ddexp(0.0, sqrt(2.0))
  }
}
"


data1_jags = list(y = X[, "class1_dummy"],
                  Sepal.Length = X[,"Sepal.Length"],
                  Sepal.Width = X[, "Sepal.Width"],
                  Petal.Length = X[, "Petal.Length"],
                  Petal.Width = X[, "Petal.Width"] )

params = c("int", "b")

model1 = jags.model(textConnection(mod1_string), data=data1_jags, n.chains = 3)
update(model1, 1e3)

mod1_sim = coda.samples(model=model1, variable.names = params, n.iter = 5e3)
mod1_csim = as.mcmc(do.call(rbind, mod1_sim))
```

Gelman and Rubin's convergence diagnostic shows that model is converged. Even though there is strong autocorrelation, the effective sample size is sufficiently large.

The penalized deviance of DIC is as below.

```
## Mean deviance:  4.315
## penalty 2.911
## Penalized deviance: 7.226
```

The dense plot suggests that the mean of Sepal.Width and Petal.Length is not equal to zero. After fitting the model with Sepal.Width and Petal.Length variables,

the penalized deviance of the DIC is higher than the first model. On the other hand, glm() of R does not converge if I fit the model with all 4 variables. Therefore I use the second prediction model.

```
## Mean deviance:  4.709
```

```
## penalty 2.553
## Penalized deviance: 7.262
```

The second model shows that 100% accuracy for classifying setosa and non-setosa.

```
(tab0.5_setosa = table(p_hat_setosa < 0.5, X[, "class1_dummy"]))
```

```
##
##              0   1
##    FALSE 100   0
##    TRUE    0  50
```

Classifying virginica and non-virginica

I followed the same procedure to get parameters for classifying virginica and non-virginica.

```
mod4_string = " model {
  for(i in 1:length(y)) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = int + b[1] * Petal.Length[i] + b[2] * Petal.Width[i]
  }

  int ~ dnorm(0.0, 1.0/25.0)
  for(j in 1:2) {
    b[j] ~ ddexp(0.0, sqrt(2.0))
  }
}
"


data4_jags = list(y = X[, "class2_dummy"],
                  Petal.Length = X[, "Petal.Length"],
                  Petal.Width = X[, "Petal.Width"]  )

params = c("int", "b")

model4 = jags.model(textConnection(mod4_string), data=data4_jags, n.chains = 3)
update(model4, 1e3)

mod4_sim = coda.samples(model=model4, variable.names = params, n.iter = 5e3)
mod4_csim = as.mcmc(do.call(rbind, mod4_sim))
```

The confusion matrix for this model shows 95% accuracy.

```
##
##              0   1
##    FALSE 97   4
##    TRUE    3  46
```

Lastly, I checked the classification performance for virginica data. The accuracy of virginica classification is 100%

```
versicolor_df = iris_normalized[iris_normalized$Species == 'versicolor', ]
```

```
X_versicolor = as.matrix( versicolor_df[2:3] )

X_pred_setosa = coef_setosa["int"] + X_versicolor %*% coef_setosa[1:2]
```

```
p_hat_setosa = 1.0 / (1.0 + exp(X_pred_setosa))
table(p_hat_setosa < 0.5, versicolor_df[, "class1_dummy"])
```

```
##
##              0
##     FALSE 50
```

```
X_versicolor = as.matrix( versicolor_df[3:4] )

X_pred_virginica = coef_virginica["int"] + X_versicolor %*% coef_virginica[1:2]
p_hat_virginica = 1.0 / (1.0 + exp(X_pred_virginica))

table(p_hat_virginica > 0.5, versicolor_df[, "class2_dummy"])
```

```
##
##              0
##     FALSE 50
```

note: comparison with ordinary glm()

glm() and aic suggests different features to fit. For example, setosa and non-setosa classification, glm() suggests Sepal.Length is the most significant feature while our previous models selected Sepal.Width and Petal.Length

```
for(i in 1:4) {
  feature_name = colnames(X)[i]
  tryCatch (
    {
      model = glm(X[, "class1_dummy"] ~ X[,i], family="binomial")
      print(paste(feature_name, " Deviance :", model$deviance, "AIC", model$aic))
    },
    warning = function(w){
      print(paste(feature_name, ": ", w))
    }
  )

}
```

```
## [1] "Sepal.Length  Deviance : 71.8363992272217 AIC  75.8363992272217"
## [1] "Sepal.Width  Deviance : 123.828029803175 AIC 127.828029803175"
## [1] "Petal.Length  :  simpleWarning: glm.fit: algorithm did not converge\n"
## [1] "Petal.Width  :  simpleWarning: glm.fit: algorithm did not converge\n"
```

Conclusion

The multiclass logistic regression model performs well for classifying iris species and in this specific example, However, it still failed to perform well for predicting iris phenotypes. Since classifying iris phenotypes is essential for taxonomic classification, it would be beneficial to employ a model that can predict specific iris phenotypes. Therefore, the discriminative binary logistic regression is a better choice when there are few phenotypic variables for which to perform a classification, significant features are different between Bayesian and frequentist approach.