GLM

Jeffrey Strickland

2/7/2022

Interpreting generalized linear models (GLM) obtained through glm is similar to [interpreting conventional linear models](/post/machine-learning/linear_models/). Here, we will discuss the differences that need to be considered.

## Basics of GLMs

GLMs enable the use of linear models in cases where the response variable has an error distribution that is non-normal. Each distribution is associated with a specific canonical link function. A link function fulfills . For example, for a Poisson distribution, the canonical link function is . Estimates on the original scale can be obtained by taking the inverse of the link function, in this case, the exponential function: .

## Data preparation

We will take 70% of the airquality samples for training and 30% for testing:

data(airquality)  
ozone <- subset(na.omit(airquality),   
 select = c("Ozone", "Solar.R", "Wind", "Temp"))  
set.seed(123)  
N.train <- ceiling(0.7 \* nrow(ozone))  
N.test <- nrow(ozone) - N.train  
trainset <- sample(seq\_len(nrow(ozone)), N.train)  
testset <- setdiff(seq\_len(nrow(ozone)), trainset)

## Training a GLM

For investigating the characteristics of GLMs, we will train a model, which assumes that errors are Poisson distributed.

By specifying family = "poisson", glm automatically selects the appropriate canonical link function, which is the logarithm. More information on possible families and their canonical link functions can be obtained via ?family.

model.pois <- glm(Ozone ~ Solar.R + Temp + Wind, data = ozone,   
 family = "poisson", subset = trainset)  
summary(model.pois)

##   
## Call:  
## glm(formula = Ozone ~ Solar.R + Temp + Wind, family = "poisson",   
## data = ozone, subset = trainset)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.0451 -2.4138 -0.8169 1.4265 8.7946   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.690227 0.218192 3.163 0.00156 \*\*   
## Solar.R 0.001815 0.000239 7.593 3.13e-14 \*\*\*  
## Temp 0.043500 0.002295 18.956 < 2e-16 \*\*\*  
## Wind -0.084244 0.005960 -14.134 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 1979.57 on 77 degrees of freedom  
## Residual deviance: 530.51 on 74 degrees of freedom  
## AIC: 951.97  
##   
## Number of Fisher Scoring iterations: 4

In terms of the GLM summary output, there are the following differences to the output obtained from the lm summary function:

* Deviance (deviance of residuals / null deviance / residual deviance)
* Other outputs: dispersion parameter, AIC, Fisher Scoring iterations

Moreover, the prediction function of GLMs is also a bit different. We will start with investigating the deviance.

## Deviance residuals

We already know [residuals](/post/machine-learning/linear_models/) from the lm function. But what are deviance residuals? In ordinary least-squares, the residual associated with the -th observation is defined as

where is the prediction function of the fitted model.

For GLMs, there are [several ways for specifying residuals](https://stackoverflow.com/questions/2531489/understanding-glmresiduals-and-residglm). To understand deviance residuals, it is worthwhile to look at the other types of residuals first. For this, we define a few variables first:

expected <- ozone$Ozone[trainset]  
g <- family(model.pois)$linkfun # log function  
g.inv <- family(model.pois)$linkinv # exp function  
estimates.log <- model.pois$linear.predictors # estimates on log scale  
estimates <- fitted(model.pois) # estimates on response scale (exponentiated)  
all.equal(g.inv(estimates.log), estimates)

## [1] TRUE

We will cover four types of residuals: response residuals, working residuals, Pearson residuals, and, deviance residuals. There is also another type of residual called *partial residual*, which is formed by determining residuals from models where individual features are excluded. This residual is not discussed here.

### Response residuals

For type = "response", the *conventional* residual on the response level is computed, that is,

This means that the fitted residuals are transformed by taking the inverse of the link function:

# type = "response"  
res.response1 <- residuals(model.pois, type = "response")  
res.response2 <- expected - estimates  
all.equal(res.response1, res.response2)

## [1] TRUE

### Working residuals

For type = "working", the residuals are normalized by the estimates :

# type = "working"  
res.working1 <- residuals(model.pois, type="working")  
res.working2 <- (expected - estimates) / estimates  
all.equal(res.working1, res.working2)

## [1] TRUE

### Pearson residuals

For type = "pearson", the Pearson residuals are computed. They are obtained by normalizing the residuals by the square root of the estimate:

# type = "pearson"  
res.pearson1 <- residuals(model.pois, type="pearson")  
res.pearson2 <- (expected - estimates) / sqrt(estimates)  
all.equal(res.pearson1, res.pearson2)

## [1] TRUE

### Deviance residuals

Deviance residuals are defined by the deviance. The deviance of a model is given by

where

* is the outcome
* is the estimate of the model
* and are the parameters of the fitted *saturated* and *proposed models*, respectively. A saturated model has as many parameters as it has training points, that is, . Thus, it has a perfect fit. The proposed model can be the any other model.
* is the likelihood of data given the model

The deviance indicates the extent to which the likelihood of the saturated model exceeds the likelihood of the proposed model. If the proposed model has a good fit, the deviance will be small. If the proposed model has a bad fit, the deviance will be high. For example, for the Poisson model, the deviance is

In R, the deviance residuals represent the contributions of individual samples to the deviance . More specifically, they are defined as the signed [square roots of the unit deviances](https://www.youtube.com/watch?v=JC56jS2gVUE). Thus, the deviance residuals are analogous to the conventional residuals: when they are squared, we obtain the sum of squares that we use for assessing the fit of the model. However, while the sum of squares is the residual sum of squares for linear models, for GLMs, this is the deviance.

How does such a deviance look like in practice? For example, for the Poisson distribution, the deviance residuals are defined as:

Let us verify this in R:

# type = "deviance"  
res.dev1 <- residuals(model.pois, type = "deviance")  
res.dev2 <- residuals(model.pois)  
poisson.dev <- function (y, mu)   
 # unit deviance  
 2 \* (y \* log(ifelse(y == 0, 1, y/mu)) - (y - mu))  
res.dev3 <- sqrt(poisson.dev(expected, estimates)) \*   
 ifelse(expected > estimates, 1, -1)  
all.equal(res.dev1, res.dev2, res.dev3)

## [1] TRUE

Note that, for ordinary least-squares models, the [deviance residual is identical to the conventional residual](http://people.stat.sfu.ca/~raltman/stat402/402L11.pdf).

### Deviance residuals in practice

We can obtain the deviance residuals of our model using the residuals function:

summary(residuals(model.pois))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -4.0451 -2.4138 -0.8169 -0.1840 1.4265 8.7946

Since the median deviance residual is close to zero, this means that our model is not biased in one direction (i.e. the out come is neither over- nor underestimated).

## Null and residual deviance

Since we have already introduced the deviance, [understanding the null and residual deviance](https://stats.stackexchange.com/questions/108995/interpreting-residual-and-null-deviance-in-glm-r) is not a challenge anymore. Let us repeat the definition of the deviance once again:

The null and residual deviance differ in :

* Null deviance: refers to the null model (i.e. an intercept-only model)
* Residual deviance: refers to the trained model

How can we interpret these two quantities?

* Null deviance: A low null deviance implies that the data can be modeled well merely using the intercept. If the null deviance is low, you should consider using few features for modeling the data.
* Residual deviance: A low residual deviance implies that the model you have trained is appropriate. Congratulations!

### Null deviance and residual deviance in practice

Let us investigate the null and residual deviance of our model:

paste0(c("Null deviance: ", "Residual deviance: "),  
 round(c(model.pois$null.deviance, deviance(model.pois)), 2))

## [1] "Null deviance: 1979.57" "Residual deviance: 530.51"

These results are somehow reassuring. First, the null deviance is high, which means it makes sense to use more than a single parameter for fitting the model. Second, the residual deviance is relatively low, which indicates that the log likelihood of our model is close to the log likelihood of the saturated model.

However, for a well-fitting model, [the residual deviance should be close to the degrees of freedom](https://stats.stackexchange.com/questions/37732/when-someone-says-residual-deviance-df-should-1-for-a-poisson-model-how-appro) (74), which is not the case here. For example, this could be a result of overdispersion where the variation is greater than predicted by the model. This can happen for a Poisson model when the actual variance exceeds the assumed mean of .

## Other outputs of the summary function

Here, I deal with the other outputs of the GLM summary fuction: the dispersion parameter, the AIC, and the statement about Fisher scoring iterations.

### Dispersion parameter

Dispersion (variability/scatter/spread) simply indicates whether a distribution is wide or narrow. The GLM function can use a dispersion parameter to model the variability.

However, for likelihood-based model, the dispersion parameter is always fixed to 1. It is adjusted only for methods that are based on quasi-likelihood estimation such as when family = "quasipoisson" or family = "quasibinomial". These methods are particularly suited for dealing with overdispersion.

### AIC

The Akaike information criterion (AIC) is an information-theoretic measure that describes the quality of a model. It is defined as

where is the number of model parameters and is the maximum of the likelihood function. A model with a low AIC is characterized by low complexity (minimizes ) and a good fit (maximizes ).

### Fisher scoring iterations

The information about *Fisher scoring iterations* is just verbose output of [iterative weighted least squares](https://hal.archives-ouvertes.fr/hal-01577698). A high number of iterations may be a cause for concern indicating that the algorithm is not converging properly.

## The prediction function of GLMs

The GLM predict function has some peculiarities that should be noted.

### The type argument

Since models obtained via lm do not use a linker function, the predictions from predict.lm are always on the scale of the outcome (except if you have transformed the outcome earlier). For predict.glm this is not generally true. Here, the type parameter determines the scale on which the estimates are returned. The following two settings are important:

* type = "link": the default setting returns the estimates on the scale of the link function. For example, for Poisson regression, the estimates would represent the logarithms of the outcomes. Given the estimates on the link scale, you can transform them to the estimates on the response scale by taking the inverse link function.
* type = "response": returns estimates on the level of the outcomes. This is the option you need if you want to evaluate predictive performance.

Let us see how the returned estimates differ depending on the type argument:

# prediction on link scale (log)  
pred.l <- predict(model.pois, newdata = ozone[testset, ])  
summary(pred.l)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.219 3.118 3.680 3.603 4.136 4.832

# prediction on respone scale  
pred.r <- predict(model.pois, newdata = ozone[testset, ], type = "response")  
summary(pred.r)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 9.20 22.60 39.63 44.25 62.58 125.46

Using the link and inverse link functions, we can transform the estimates into each other:

link <- family(model.pois)$linkfun # link function: log for Poisson  
ilink <- family(model.pois)$linkinv # inverse link function: exp for Poisson  
all.equal(ilink(pred.l), pred.r)

## [1] TRUE

all.equal(pred.l, link(pred.r))

## [1] TRUE

There is also the type = "terms" setting but this one is rarely used an also available in predict.lm.

### Obtaining confidence intervals

The predict function of GLMs does not support the output of confidence intervals via interval = "confidence" as for predict.lm. We can still obtain confidence intervals for predictions by accessing the standard errors of the fit by predicting with se.fit = TRUE:

predict.confidence <- function(object, newdata, level = 0.95, ...) {  
 if (!is(object, "glm")) {  
 stop("Model should be a glm")  
 }  
 if (!is(newdata, "data.frame")) {  
 stop("Plase input a data frame for newdata")  
 }  
 if (!is.numeric(level) | level < 0 | level > 1) {  
 stop("level should be numeric and between 0 and 1")  
 }  
 ilink <- family(object)$linkinv  
 ci.factor <- qnorm(1 - (1 - level)/2)  
 # calculate CIs:  
 fit <- predict(object, newdata = newdata, level = level,   
 type = "link", se.fit = TRUE, ...)  
 lwr <- ilink(fit$fit - ci.factor \* fit$se.fit)  
 upr <- ilink(fit$fit + ci.factor \* fit$se.fit)  
 df <- data.frame("fit" = ilink(fit$fit), "lwr" = lwr, "upr" = upr)  
 return(df)  
}

Using this function, we get the following confidence intervals for the Poisson model:

conf.df <- predict.confidence(model.pois, ozone[testset,])  
head(conf.df, 2)

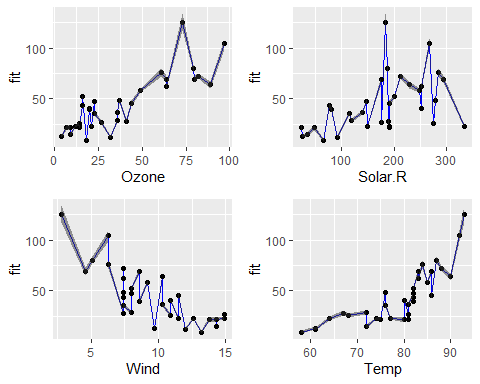
## fit lwr upr  
## 1 27.83060 25.59152 30.26559  
## 2 28.85877 26.91558 30.94225

Using the confidence data, we can create a function for plotting the confidence of the estimates in relation to individual features:

plot.confidence <- function(df, feature) {  
 library(ggplot2)  
 p <- ggplot(df, aes\_string(x = feature,   
 y = "fit")) +   
 geom\_line(colour = "blue") +   
 geom\_point() +   
 geom\_ribbon(aes(ymin = lwr, ymax = upr),   
 alpha = 0.5)   
 return(p)  
}  
plot.confidence.features <- function(data, features) {  
 plots <- list()  
 for (feature in features) {  
 p <- plot.confidence(data, feature)  
 plots[[feature]] <- p  
 }  
 library(gridExtra)  
 #grid.arrange(plots[[1]], plots[[2]], plots[[3]])  
 do.call(grid.arrange, plots)  
}

Using these functions, we can generate the following plot:

data <- cbind(ozone[testset,], conf.df)  
plot.confidence.features(data, colnames(ozone))



## Where to go from here?

Having covered the fundamentals of GLMs, you may want to dive deeper into their practical application by taking a look at [this post where I investigate different types of GLMs for improving the prediction of ozone levels](/post/machine-learning/improving_ozone_prediction/).