STATS 266 Handout - Rmarkdown and LaTeX

Qi Wang

2025-04-29

Contents

1 Generalized Linear Models (GLM)

Generalized Linear Models (GLMs) extend traditional linear models by allowing the response variable to have a distribution other than the normal, such as binomial or Poisson. A GLM consists of three main components:

1. **Random Component**: Specifies the probability distribution of the response variable (e.g., Binomial, Poisson, Gaussian), which must be a member of the exponential family.
2. **Systematic Component**: A linear predictor, typically represented as $\eta = X\beta$, where X is the design matrix of covariates and β are the coefficients.
3. **Link Function**: A function $g(\cdot)$ that connects the mean of the distribution $\mu = \mathbb{E}[Y]$ to the linear predictor, i.e., $g(\mu) = \eta$.

The choice of link function depends on the distribution of the response variable. Common link functions include:

- **Logit link** for binary outcomes (logistic regression): $\log\left(\frac{\mu}{1-\mu}\right)$
- **Log link** for count data (Poisson regression): $\log(\mu)$
- **Identity link** for continuous normal outcomes: μ itself

Understanding the link function is crucial: it transforms a restricted range of the mean (such as [0,1] for probabilities) to the whole real line, making linear modeling appropriate.

2 Logistic Regression (Binomial GLM)

Logistic regression models a binary outcome using a logit link function. It models the **log-odds** of the probability of success as a linear combination of the predictors.

Model Specification:

$$Y_i \sim \text{Binomial}(1, \pi_i), \quad \log\left(\frac{\pi_i}{1-\pi_i}\right) = X_i\beta$$

Likelihood Function:

$$L(\beta) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

```r
# Simulate data for logistic regression
set.seed(123)
n <- 100
x <- rnorm(n)
z <- 1 + 2 * x
p <- 1 / (1 + exp(-z))
```

```r
y <- rbinom(n, size = 1, prob = p)
logistic_data <- data.frame(x = x, y = y)

# Fit logistic regression model
logit_fit <- glm(y ~ x, data = logistic_data, family = binomial)

# Summary of the model
summary(logit_fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial, data = logistic_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0434     0.3063   3.407 0.000657 ***
## x             2.4414     0.5241   4.658  3.2e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 126.836  on 99  degrees of freedom
## Residual deviance:  81.452  on 98  degrees of freedom
## AIC: 85.452
##
## Number of Fisher Scoring iterations: 6
```

Interpretation:

- The coefficient of `x` represents the change in the **log-odds** for a one-unit increase in `x`.
- Exponentiating the coefficient gives the **odds ratio** associated with a one-unit increase in `x`.
- The intercept represents the log-odds when `x = 0`.

3 Poisson Regression

Poisson regression models count data and assumes the counts follow a Poisson distribution. It uses a **log link** to model the log of the expected count as a linear combination of predictors.

Model Specification:

$$Y_i \sim \text{Poisson}(\lambda_i), \quad \log(\lambda_i) = X_i\beta$$

Likelihood Function:

$$L(\beta) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}$$

```r
# Simulate data for Poisson regression
set.seed(456)
x <- rnorm(n)
lambda <- exp(0.5 + 0.3 * x)
y <- rpois(n, lambda = lambda)
poisson_data <- data.frame(x = x, y = y)

# Fit Poisson regression model
poisson_fit <- glm(y ~ x, data = poisson_data, family = poisson)

# Summary of the model
summary(poisson_fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = poisson, data = poisson_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.34259    0.08814   3.887 0.000101 ***
## x            0.31998    0.08200   3.902 9.53e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 149.62  on 99  degrees of freedom
## Residual deviance: 134.18  on 98  degrees of freedom
## AIC: 321.12
##
## Number of Fisher Scoring iterations: 5
```

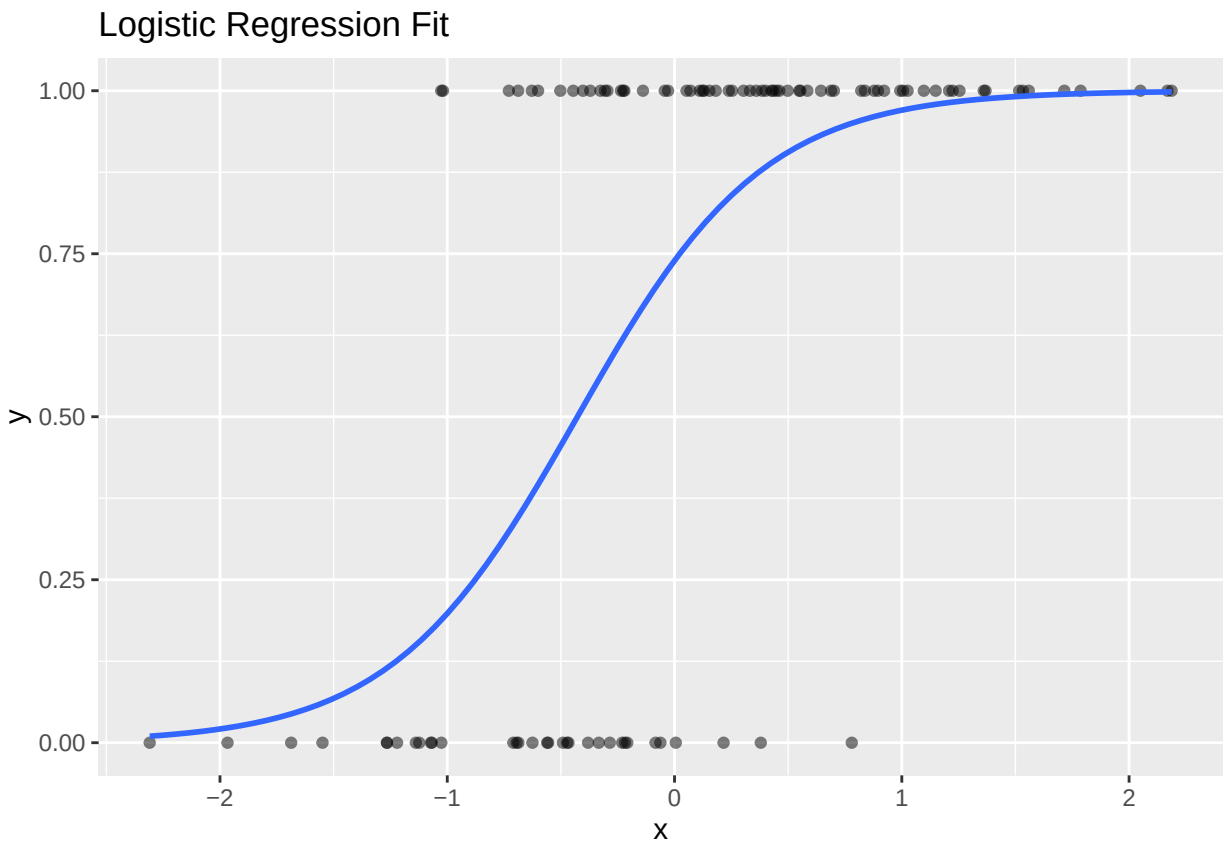
Interpretation:

- The coefficient for `x` represents the change in the **log of the expected count** for a one-unit increase in `x`.
- Exponentiating the coefficient gives the **multiplicative effect** on the expected count for a one-unit increase in `x`.

4 Visualizing the Fits

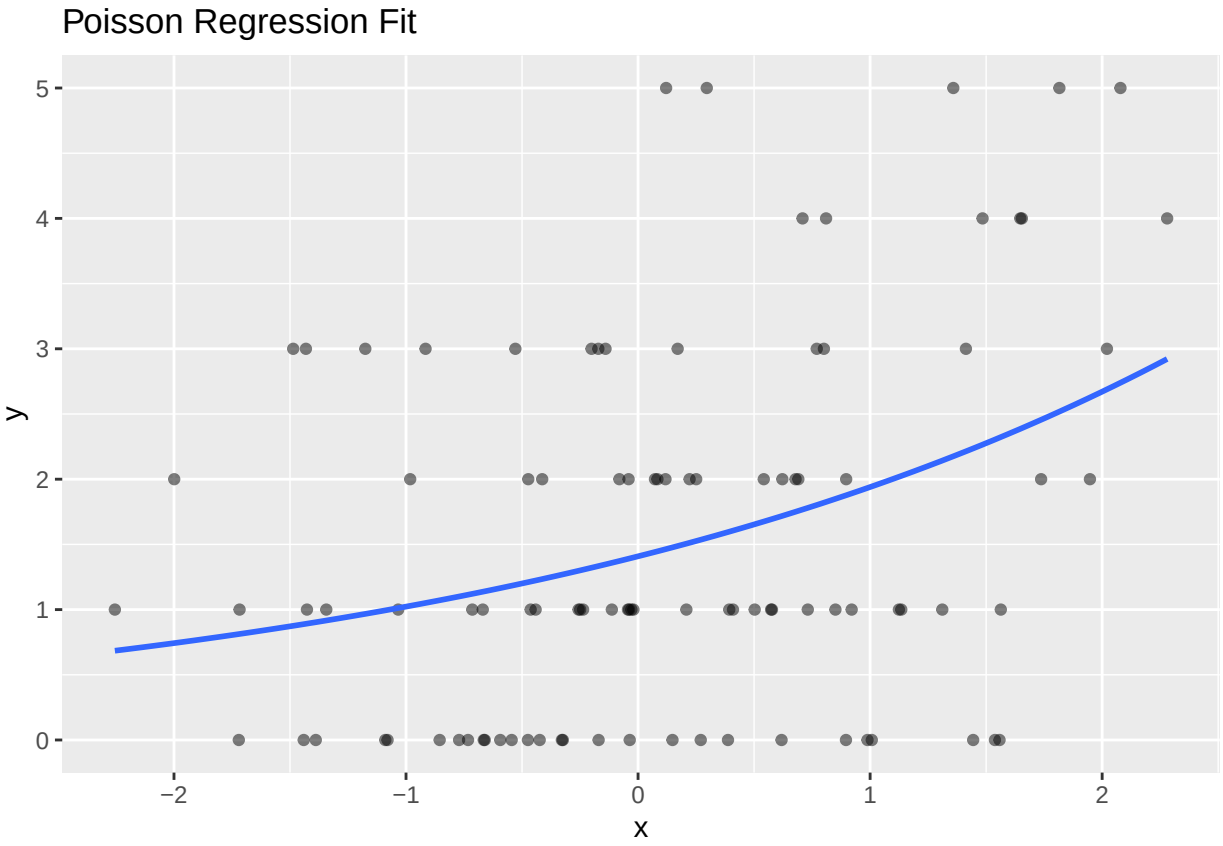
```r
library(ggplot2)

# Logistic Regression Fit
p1 <- ggplot(logistic_data, aes(x = x, y = y)) +
  geom_point(alpha = 0.5) +
  stat_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE) +
  ggtitle("Logistic Regression Fit")

# Poisson Regression Fit
p2 <- ggplot(poisson_data, aes(x = x, y = y)) +
  geom_point(alpha = 0.5) +
  stat_smooth(method = "glm", method.args = list(family = "poisson"), se = FALSE) +
  ggtitle("Poisson Regression Fit")

p1
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
p2
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



GLMs are powerful because they generalize linear modeling to a wide range of data types through flexible distributional assumptions and link functions. Proper understanding of the link function, the likelihood structure, and the nature of the response variable is essential for correct model specification and interpretation.

5 Penalized Regression: Ridge and Lasso

Penalized regression techniques are useful when dealing with high-dimensional data, multicollinearity, or when regularization is desired to prevent overfitting.

5.1 Ridge Regression

Ridge regression adds an ℓ_2 penalty (squared magnitude of coefficients) to the loss function.

For linear regression:

$$\text{Minimize} \quad \sum_{i=1}^{n}(y_i - X_i\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

where $\lambda \geq 0$ controls the strength of the penalty.

For GLM (general case), the penalized log-likelihood becomes:

$$\text{Maximize} \quad \ell(\beta) - \lambda \sum_{j=1}^{p} \beta_j^2$$

5.2 Lasso Regression

Lasso regression adds an ℓ_1 penalty (absolute magnitude of coefficients), encouraging sparsity (some coefficients exactly zero).

For linear regression:

$$\text{Minimize} \quad \sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

For GLMs:

$$\text{Maximize} \quad \ell(\beta) - \lambda \sum_{j=1}^{p} |\beta_j|$$

5.3 Elastic Net

Elastic Net combines Ridge and Lasso penalties.

$$\text{Minimize} \quad \sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right)$$

where $0 \leq \alpha \leq 1$ controls the balance between Lasso and Ridge.

5.4 Using `glmnet` in R

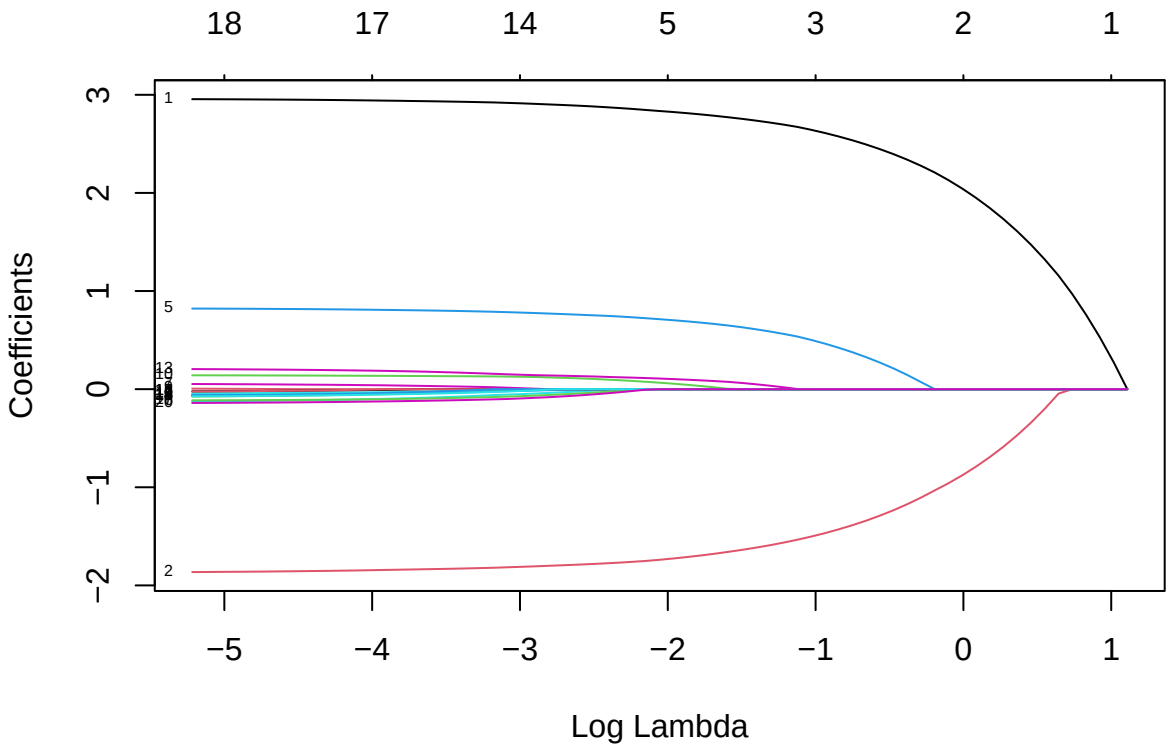
```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.1
```

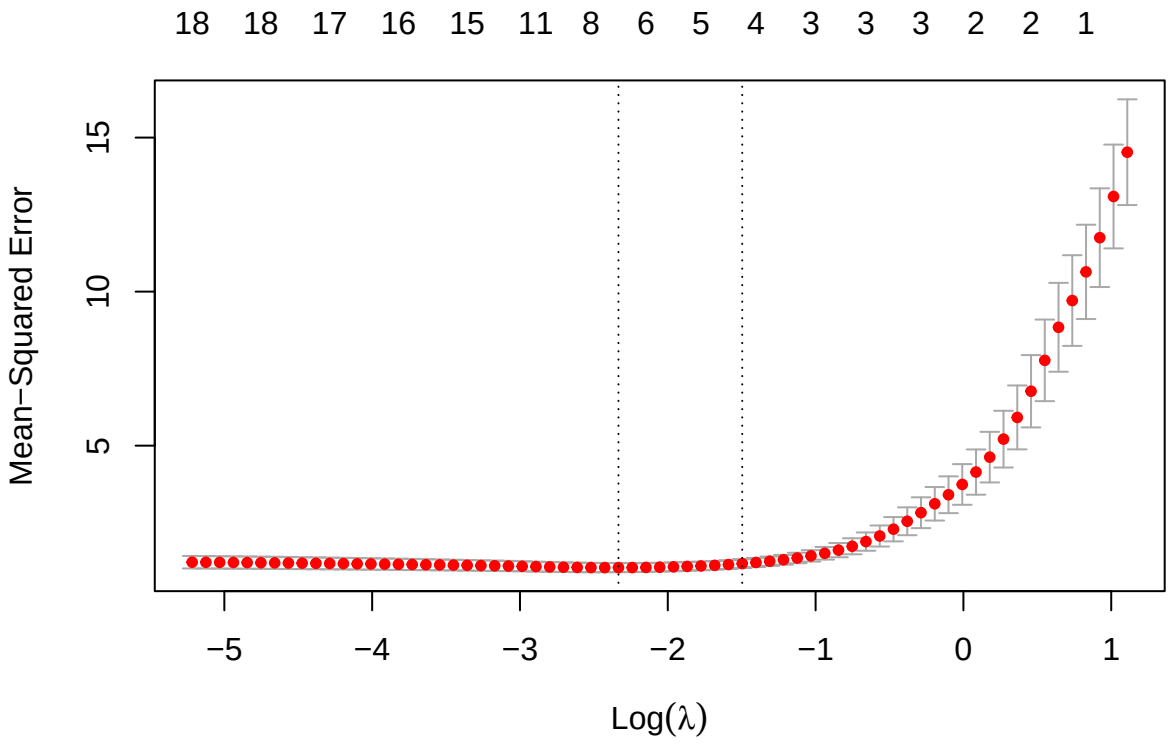
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
# Simulate some data
set.seed(789)
X <- matrix(rnorm(100*20), 100, 20)
beta <- c(3, -2, 0, 0, 1, rep(0, 15))
y <- X %*% beta + rnorm(100)

# Fit Ridge Regression
ridge_fit <- glmnet(X, y, alpha = 0) # alpha=0 for Ridge

# Fit Lasso Regression
lasso_fit <- glmnet(X, y, alpha = 1) # alpha=1 for Lasso

# Plot coefficient paths
plot(lasso_fit, xvar = "lambda", label = TRUE)
```




```r
# Cross-validation to choose optimal lambda
cv_lasso <- cv.glmnet(X, y, alpha = 1)
plot(cv_lasso)
```




```
# Best lambda
cv_lasso$lambda.min
```

```
## [1] 0.09694381
```

Notes: - `alpha = 1` corresponds to Lasso. - `alpha = 0` corresponds to Ridge. - `alpha` between 0 and 1 corresponds to Elastic Net. - For GLM families (e.g., binomial for logistic regression, poisson), set the `family` argument in `glmnet` (e.g., `family = "binomial"`).