

STAT3612 Lecture 3: Generalized Linear Models

Dr. Aijun Zhang

28 January - 14 February 2019

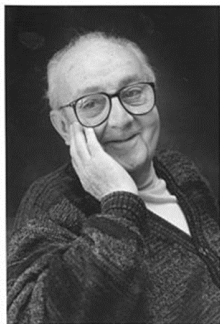


Department of 統計及精算學系
Statistics & Actuarial Science

Table of Contents

- 1 Generalized Linear Models
- 2 Linear Regression
- 3 Logistic Regression
- 4 Softmax Regression

George Box



George Box (1919–2013)

[Wikipedia](#)

- “One of the great statistical minds of the 20th century”
- Most famous quote: *“Essentially, all models are wrong, but some are useful.”*
- Nate Silver (2012, *The Signal and the Noise*): *What Box meant is that all models are simplifications of the universe, as they must necessarily be. As another mathematician said, “The best model of a cat is a cat.”*
- Norbert Wiener (1945, *Philosophy of Science*): *The best material model of a cat is another, or preferably the same, cat.*
- Another quote by Box: *“Statisticians, like artists, have the bad habit of falling in love with their models.”*

Generalized Linear Models

- Under supervised learning settings, consider the regression problem with the feature vector $X \in \mathbb{R}^{p-1}$ and the response $Y \in \mathbb{R}$.
- Denote by $\mu(\mathbf{x}) = \mathbb{E}[Y|X = \mathbf{x}]$ the conditional mean of Y given $X = \mathbf{x}$.
- A generalized linear model (GLM) takes the general form

$$g[\mu(\mathbf{x})] = \beta_0 + \beta_1 x_1 + \cdots + \beta_{p-1} x_{p-1} \equiv \eta(\mathbf{x})$$

where $g : \mathbb{R} \rightarrow \mathbb{R}$ is a strictly monotonic link function, and $\eta(\mathbf{x})$ is the linear predictor involving the intercept β_0 and the coefficients $\{\beta_j\}_{j \in [p-1]}$.

- The choice of link function g depends on the types of the response variable, e.g. Gaussian, Binomial, Multinomial, Poisson, etc.

Generalized Linear Models: Link Functions

- When Y is quantitative and ideally follows the Gaussian (i.e. Normal) distribution, we simply use the **identity** link:

$$\eta \leftarrow g[\mu] = \mu \quad \text{(Linear regression)}$$

- When Y is binary (e.g. $\{0, 1\}$), $\mu(\mathbf{x}) = \mathbb{P}(Y = 1|X = \mathbf{x})$, which equals the success probability of the binomial distribution. We use the **logit** link:

$$\eta \leftarrow g[\mu] = \log\left(\frac{\mu}{1 - \mu}\right) \quad \text{(Logistic regression)}$$

- When Y is multcategory (K unordered classes), we use the **multinomial logit** (inverse) link:

$$\mathbb{P}(Y = k|X = \mathbf{x}) = \frac{e^{\eta_k(\mathbf{x})}}{\sum_{l=1}^K e^{\eta_l(\mathbf{x})}} \quad \text{(Softmax regression)}$$

where each class gets its own linear prediction $\eta_l(\mathbf{x})$ for $l = 1, \dots, K$.

Generalized Linear Models: Link Functions

- When Y represents counts $\{0, 1, 2, \dots\}$ and follows the Poisson distribution

$$\mathbb{P}(Y = k | X = \mathbf{x}) = \frac{\lambda(\mathbf{x})^k}{k!} e^{-\lambda(\mathbf{x})}, \quad k = 0, 1, 2, \dots$$

we have that $\mu(\mathbf{x}) = \lambda(\mathbf{x}) \geq 0$, and use the natural **log** link:

$$\eta \leftarrow g[\mu] = \log(\mu) \quad \text{(Poisson regression)}$$

- When Y represents the time-to-failure response, the hazard function

$$\lambda(Y = t | X = \mathbf{x}) = \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \mathbb{P}(t \leq Y \leq t + \Delta t | X = \mathbf{x})$$

can be treated as the time-varying rates for a nonhomogeneous Poisson process. The Cox's proportional hazards (CoxPH) model of the form $\lambda(t|\mathbf{x}) = \lambda_0(t)e^{\beta^T \mathbf{x}}$ corresponds to the **relative risk log** link:

$$\eta(t) \leftarrow \log \left(\frac{\lambda(t|\mathbf{x})}{\lambda_0(t)} \right) \quad \text{(Survival regression)}$$

Generalized Linear Models: Remarks



- The classical GLMs by McCullagh and Nelder (1989) are described by an exponential family of distributions (e.g. Gaussian, Bernoulli, Poisson, and Gamma). For such models, $\mathbb{E}[Y|X = \mathbf{x}] = \mu(\mathbf{x}) = g^{-1}(\eta(\mathbf{x}))$ while the variance is typically $\text{Var}[Y|X = \mathbf{x}] = V(\mu(\mathbf{x}))$; see [Wikipedia](#).
- The introduced link functions take the canonical forms, while there also exist other link functions (e.g. probit, cloglog for the binomial family).
- For GLMs, we can perform statistical inference: maximum likelihood estimation, hypothesis testing (e.g. p -value), and model diagnostics.
- The GLMs are interpretable, i.e. the model coefficients can be interpreted with practical language.
- Only the linear regression model admits the closed-form solution. The GLMs are usually estimated by Newton-Raphson method with Fisher scoring, or the iteratively reweighted least squares (IRLS).

Table of Contents

- 1 Generalized Linear Models
- 2 Linear Regression**
- 3 Logistic Regression
- 4 Softmax Regression

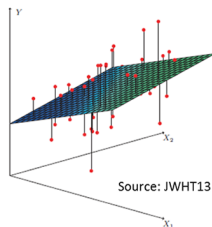
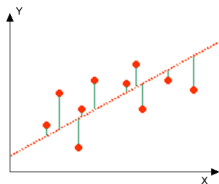
Linear Regression Model

- Given the n -sample observations represented by $X \in \mathbb{R}^{n \times p}$ (including the first column of ones) and $\mathbf{y} \in \mathbb{R}^n$, the linear model takes the form

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

- The unknown vector of parameters $\boldsymbol{\beta} \in \mathbb{R}^p$ is estimated by minimizing the mean squared error (MSE):

$$\min_{\boldsymbol{\beta}} \text{MSE}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 = \frac{1}{n} \|\mathbf{y} - X\boldsymbol{\beta}\|^2$$



Least Squares Estimation

- Differentiating MSE w.r.t. β and setting to zero, we have the **normal equation**:

$$X^T X \beta = X^T y \quad (1)$$

- When $X^T X$ is invertible, we obtain the least squares estimator (LSE):

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (2)$$

- The best linear unbiased prediction (BLUP) for y is given by

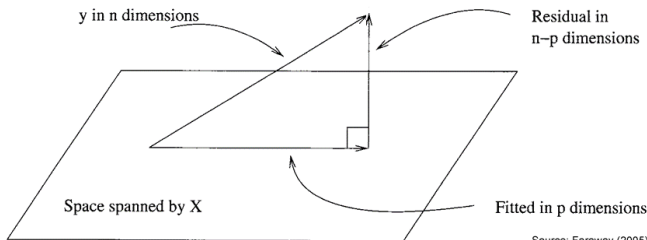
$$\hat{y} = X(X^T X)^{-1} X^T y = H y \quad (3)$$

where H is called the hat matrix and it is an orthogonal projector to the space spanned by X .

Goodness-of-fit Statistic

The percentage of variance explained (a.k.a. coefficient of determination):

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\text{SSE}}{\text{SST}} \in [0, 1] \quad (4)$$



ANOVA Test

Null hypothesis (that corresponds to the overall mean model $y = \mu + \varepsilon$):

$$H_0 : \beta_1 = \dots = \beta_{p-1} = 0$$

Testing by the F -statistic:

$$F = \frac{(\text{SST} - \text{SSE})/(p - 1)}{\text{SSE}/(n - p)} \sim F_{p-1, n-p}$$

where the null $F_{p-1, n-p}$ distribution determines a critical value or p -value.

Source	Degrees of freedom	Sum of Squares	Mean Square	F
Regression	$p - 1$	SSR	$\text{SSR}/(p - 1)$	$\frac{\text{SSR}/(p - 1)}{\text{SSE}/(n - p)}$
Residual	$n - p$	SSE	$\text{SSE}/(n - p)$	
Total	$n - 1$	SST		

Wald Test

Null hypothesis on a single parameter: $H_0 : \beta_j = 0$

Testing by the t -statistic:

$$t = \frac{\hat{\beta}_j}{\text{se}(\hat{\beta}_j)} \sim t_{n-p}$$

(or equivalently $F = t^2 \sim F_{1,n-p}$). It is straightforward to construct the confidence interval for β_j between the bounds

$$\hat{\beta}_j \pm t_{1-\alpha/2, n-p} \text{se}(\hat{\beta}_j).$$

Note that the variance σ^2 and the standard error of $\hat{\beta}_j$ can be estimated by

$$\hat{\sigma}^2 = \frac{\text{SSE}}{n-p}, \quad \text{se}(\hat{\beta}_j) = \hat{\sigma} \sqrt{(\mathbf{X}^T \mathbf{X})_{(j+1)(j+1)}^{-1}}$$

Demo Output

```
import statsmodels.api as sm

X1 = sm.add_constant(X)
lm = sm.OLS(y,X1).fit()
print(lm.summary())
```

OLS Regression Results

```
=====
Dep. Variable:          y      R-squared:                0.938
Model:                  OLS    Adj. R-squared:           0.937
Method:                 Least Squares    F-statistic:              736.9
Date:                   Thu, 31 Jan 2019    Prob (F-statistic):       6.20e-88
Time:                   15:48:40    Log-Likelihood:           36.809
No. Observations:      150    AIC:                      -65.62
Df Residuals:          146    BIC:                      -53.57
Df Model:               3
Covariance Type:       nonrobust
=====
```

```
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
const        -0.2487      0.178        -1.396      0.165      -0.601      0.103
x1            -0.2103      0.048        -4.426      0.000      -0.304     -0.116
x2             0.2288      0.049         4.669      0.000       0.132      0.326
x3             0.5261      0.024        21.536      0.000       0.478      0.574
=====
```

```
=====
Omnibus:                 5.603    Durbin-Watson:           1.577
Prob(Omnibus):           0.061    Jarque-Bera (JB):        6.817
Skew:                    0.222    Prob(JB):                 0.0331
Kurtosis:                 3.945    Cond. No.                 90.0
=====
```

Model Diagnostics

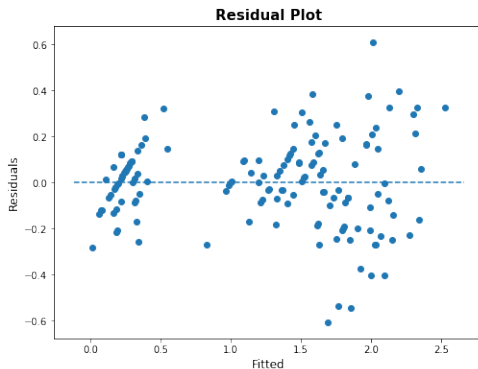
Be aware of the potential problems with the linear regression model:

- Problem with the response-feature relationship: **non-linearity**
- Problem with the error assumption: **non-normality, heteroscedasticity**
- Problem with the observations: **outliers, high-leverage points**
- Problem with the features: **collinearity, multi-collinearity**

Graphical diagnostic techniques: histogram, residual plot, influence plot ...

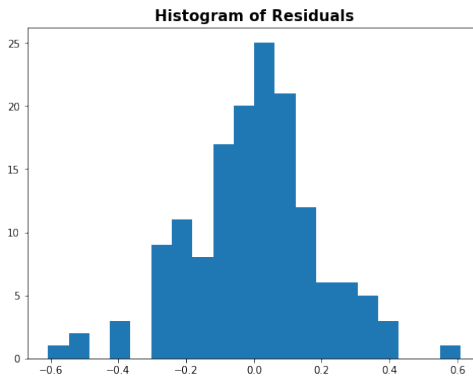
Model Diagnostics: Residual plot

Compute the residuals $\hat{\varepsilon}_i = \hat{y}_i - y_i$ for $i = 1, \dots, n$; then plot them against the fitted values.



- Check if there is any non-linear trend (non-linearity);
- Check if there is non-constant variance (heteroscedasticity).

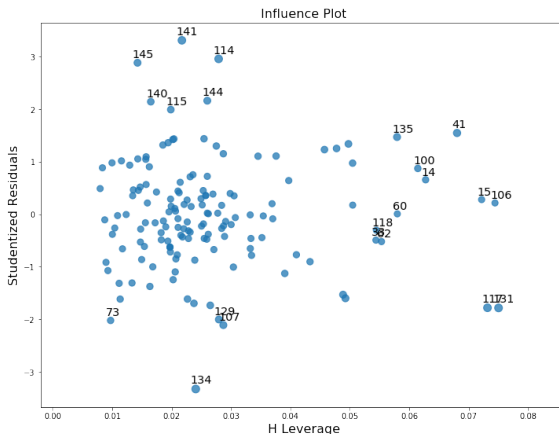
Model Diagnostics: Histogram of Residuals



- Check if there residuals are normally distributed. (Also, QQ plot)

Model Diagnostics: Influence Plot

- Leverage scores: $h_i = H_{ii}$ (hat matrix diagonal) for checking influence
- Studentized residuals: $r_i = \frac{\hat{\varepsilon}_i}{\hat{\sigma}\sqrt{1-h_i}}$ for checking outlyingness



Model Diagnostics: Collinearity

- Detect collinearity (when 2 features are highly correlated) by checking the correlation matrix of the features
- Detect multi-collinearity (when 3 or more features are highly correlated) by checking the VIF (variance inflation factor):

$$\text{VIF}(\hat{\beta}_j) = \frac{1}{1 - R_{X_j|X_{-j}}^2},$$

via regression of X_j on all other features X_{-j} , repeatedly for all j .

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
VIF = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
np.round(VIF, 2)
array([204.77,  85.62,  36.71])
```

Linear Regression: Interesting Problems



- Study of least square estimator when $X^T X$ is not invertible, a typical case for high-dimensional data ($p \gg n$). Hint: Moore–Penrose inverse.
- Generalized/weighted least squares for the linear regression model with heteroscedastic or serially correlated errors
- Transformation of the response variable (e.g. through Box-Cox method for power family of transformations)
- Linear regression model for the features with measurement errors (e.g. total least squares)
- Leverage sampling for large-scale data modeling
- Iterative Hessian sketching for approximation of $X^T X$ for big data X

Table of Contents

- 1 Generalized Linear Models
- 2 Linear Regression
- 3 Logistic Regression**
- 4 Softmax Regression

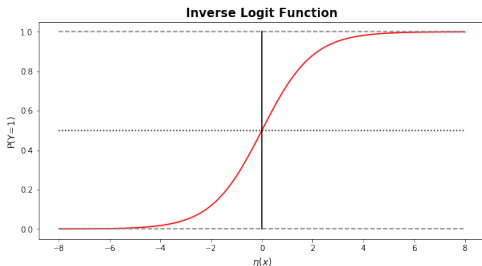
Logistic Regression

When $Y \in \{0, 1\}$, consider the GLM with the logit link function:

$$\log\left(\frac{\mu(\mathbf{x})}{1 - \mu(\mathbf{x})}\right) = \eta(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_{p-1} x_{p-1} = \boldsymbol{\beta}^T \mathbf{x}$$

The probability of $Y = 1$ is given by the inverse logit function:

$$p(\mathbf{x}) \equiv \mu(\mathbf{x}) = \frac{1}{1 + e^{-\eta(\mathbf{x})}} = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \mathbf{x}}} = \sigma(\boldsymbol{\beta}^T \mathbf{x})$$



Logistic Regression: Decision Boundary

- For binary responses, the decision boundary separates the predictions of 1's from 0's. It corresponds to $\mathbb{P}(Y = 1|\mathbf{x}) = 0.5$ or the log odds $\eta(\mathbf{x}) = 0$.
- So the decision boundary for logistic regression is given by

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_{p-1} x_{p-1} = 0.$$

- In (x_1, x_2) case, the decision boundary of abline format:

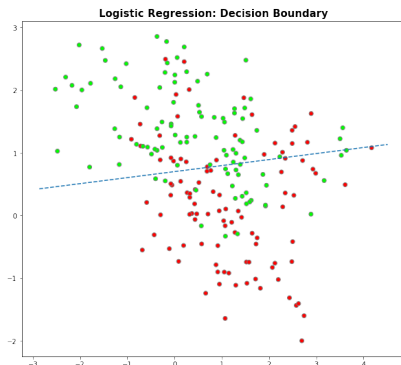
$$x_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} x_1$$

- In 2D case, we may also visualize the decision boundary by mesh grid prediction.

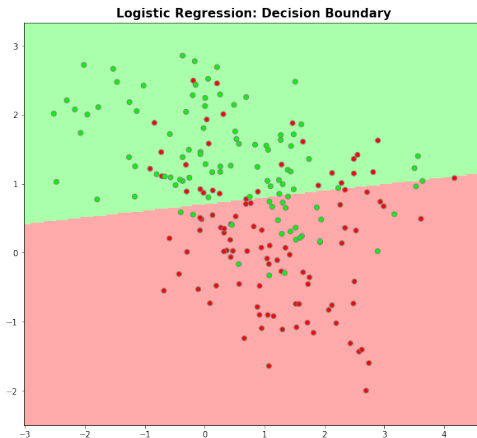
Logistic Regression: Decision Boundary

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e8)
logreg.fit(X, y)
np.round(logreg.intercept_, 4), np.round(logreg.coef_, 4)
```

```
(array([-0.978]), array([[ -0.1344,  1.3981]]))
```



Logistic Regression: Decision Boundary Visualization



Parameter Estimation

The unknown parameter β can be estimated by minimizing the negative log-likelihood (loss function) for n -sample observations:

$$\begin{aligned} L(\beta) &= -\log \prod_{i:y_i=1} p(\mathbf{x}_i) \prod_{i:y_i=0} (1 - p(\mathbf{x}_i)) \\ &= -\sum_{i=1}^n \left\{ y_i \log p(\mathbf{x}_i) + (1 - y_i) \log (1 - p(\mathbf{x}_i)) \right\} \\ &= -\sum_{i=1}^n \left\{ y_i \beta^T \mathbf{x}_i - \log (1 + e^{\beta^T \mathbf{x}_i}) \right\} \end{aligned} \quad (5)$$

Such a loss function is also known as the **cross entropy** function.

Parameter Estimation

- The optimization problem can be solved through the Newton-Raphson method in an iterative way:

$$\boldsymbol{\beta}^{\text{new}} = \boldsymbol{\beta}^{\text{old}} - \left(\frac{\partial^2 L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \Bigg|_{\boldsymbol{\beta} = \boldsymbol{\beta}^{\text{old}}}$$

based on the first-order and second-order partial derivatives.

- Since evaluations of second-order derivatives (i.e. Hessian matrix) is highly demanding when n is large, the first-order methods are often used today, which are known as stochastic gradient decent algorithms.
- We will discuss these algorithms in a later chapter on stochastic optimization.

Table of Contents

- 1 Generalized Linear Models
- 2 Linear Regression
- 3 Logistic Regression
- 4 **Softmax Regression**

Softmax Regression

$$p_k(\mathbf{x}) = \frac{\exp(\boldsymbol{\beta}_k^T \mathbf{x})}{\sum_{l=1}^K \exp(\boldsymbol{\beta}_l^T \mathbf{x})}, \quad k = 1, \dots, K$$

$$\hat{y} = \arg \max_k p_k(\mathbf{x})$$

See [here](#) for R code demonstration.

Thank You!

Q&A or Email ajzhang@hku.hk.