STAT3612 Lecture 3

Generalized Linear Models

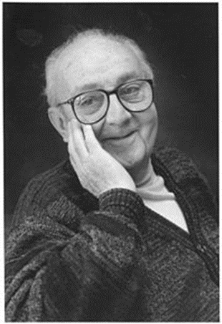
Dr. Aijun Zhang

15 September 2020

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

Table of Contents

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 (GAM)

- Under supervised settings, consider the regression problem with the feature $X \in \mathbb{R}^{p-1}$ and the response $Y \in \mathbb{R}$. Let $\mu(\boldsymbol{x}) = \mathbb{E}[Y|X = \boldsymbol{x}]$ denote the conditional mean of Y given $X = \boldsymbol{x}$.

- A generalized linear model (GLM) takes the form

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

  where $g : \mathbb{R} \to \mathbb{R}$ is a strictly monotonic link function, and $\eta(\boldsymbol{x})$ is the linear predictor involving the intercept β_0 and the coefficients $\{\beta_j\}$.

- Interpretation of GLM coefficients: a unit increase in x_j with other features fixed increases the g-transform of expected response by β_j.

- 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 continuous and follows the Gaussian (i.e. Normal) distribution, we simply use the **identity** link:

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

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

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

- When Y is multi-category (K ordinal classes), let $\gamma_j(\boldsymbol{x}) = \mathbb{P}(Y \leq j|\boldsymbol{x})$ denote the cumulative probability, we use the **ordinal logit** link:

$$\log\left(\frac{\gamma_j(\boldsymbol{x})}{1-\gamma_j(\boldsymbol{x})}\right) = \theta_j - \boldsymbol{\beta}^T\boldsymbol{x} \qquad \text{(Proportional odds model)}$$

where each class has the specified intercept θ_j.

Generalized Linear Models: Link Functions

- When Y is multi-category (K nominal classes), we use the **multinomial logit** (inverse) link:

$$\mathbb{P}(Y = k|X = \boldsymbol{x}) = \frac{e^{\eta_k(\boldsymbol{x})}}{e^{\eta_1(\boldsymbol{x})+\cdots+\eta_K(\boldsymbol{x})}} \qquad \text{(Softmax regression)}$$

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

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

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

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

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

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); see Wikipeida.
- The introduced link functions take the canonical forms, while there also exist other link functions (e.g., **logit**, probit, cloglog for the binomial and multinomial responses).
- The GLMs are intrinsically interpretable, i.e. the model coefficients can be interpreted with practical language.
- In machine learning, the linear and logistic/softmax regression models are mostly discussed.

Table of Contents

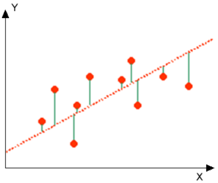
Linear Regression Model

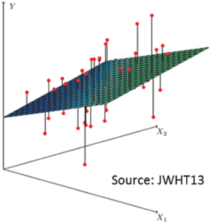
- Given the n-sample observations represented by $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ (including the first column of ones) and $\boldsymbol{y} \in \mathbb{R}^n$, the linear model takes the form

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

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

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





Source: JWHT13

Least Squares Estimation

- Differentiating MSE w.r.t. $\boldsymbol{\beta}$ and setting to zero, we have the **normal equation**:

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y} \tag{1}$$

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

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

- The best linear unbiased prediction (BLUP) for $\boldsymbol{y}$ is given by

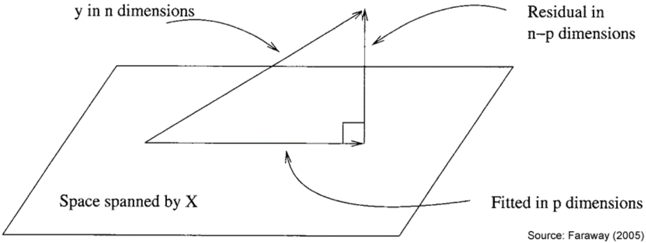
$$\hat{\boldsymbol{y}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{H}\boldsymbol{y} \tag{3}$$

  where $\boldsymbol{H}$ is called the hat matrix and it is an orthogonal projector to the space spanned by $\boldsymbol{X}$.

Goodness-of-fit Statistic

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

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



Source: Faraway (2005)

ANOVA Test

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

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

Testing by the F-statistic:

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

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	$\mathrm{SSR}/(p-1)$	$\frac{\mathrm{SSR}/(p-1)}{\mathrm{SSE}/(n-p)}$
Residual	$n-p$	SSE	$\mathrm{SSE}/(n-p)$	
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

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

Demo Output

```python
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

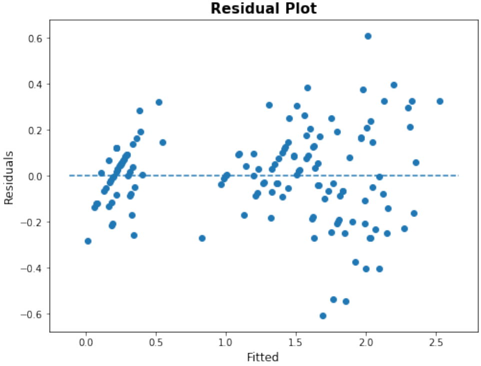
Be aware of the potential problems with the linear regression model:

- 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 ...

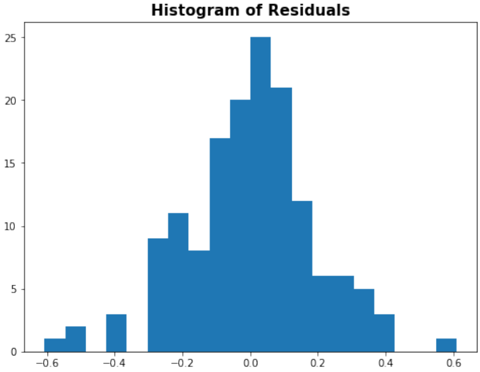
Model Diagnostics: Residual plot

Compute the residuals $\hat{\varepsilon}_i = \hat{y}_i - y_i$, and plot them against the fitted values.



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

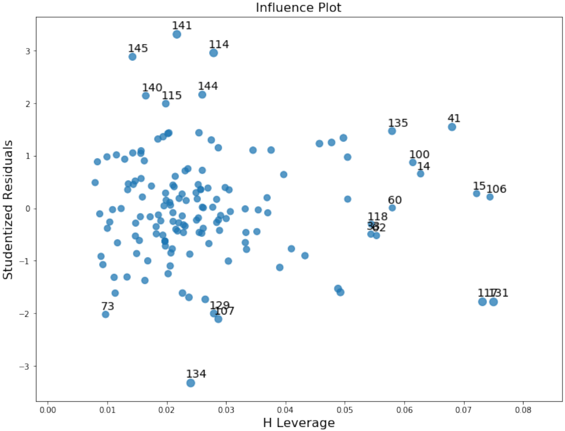
Model Diagnostics: Histogram of Residuals



- 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):

$$\mathrm{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{\boldsymbol{X}_j|\boldsymbol{X}_{-j}}},$$

via regression of $\boldsymbol{X}_j$ on all other features $\boldsymbol{X}_{-j}$, 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])
```

Table of Contents

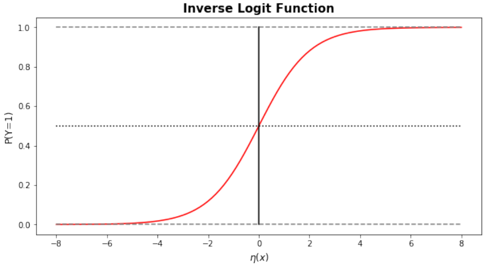
Logistic Regression

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

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

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

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



Logistic Regression: Decision Boundary

- For binary responses, the decision boundary separates the predictions of 1's from 0's. It corresponds to $\mathbb{P}(Y = 1|\boldsymbol{x}) = 0.5$ or the log odds $\eta(\boldsymbol{x}) = 0$.
- 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:

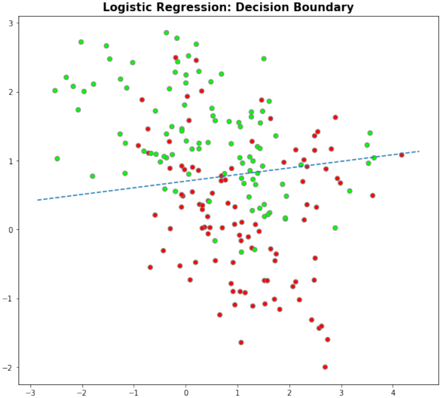
$$x_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} x_1$$

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

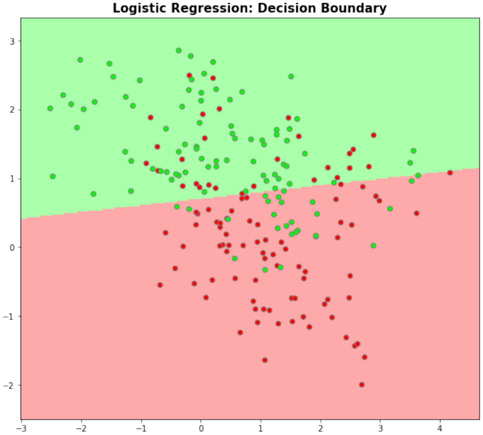
Logistic Regression: Decision Boundary

```
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 $\boldsymbol{\beta}$ can be estimated by minimizing the negative log-likelihood (loss function) for n-sample observations:

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

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

Logistic Regression: 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 (SGD) algorithms.

Logistic Regression: Be Careful in Python

```
from sklearn.linear_model import LogisticRegression

X = DataX.iloc[:,0:2]
y = DataX.iloc[:,2]
logreg = LogisticRegression()
logreg.fit(X, y)
np.round(logreg.intercept_, 4), np.round(logreg.coef_, 4)
```

```
(array([-0.8496]), array([[-0.1586,  1.293 ]]))
```

```
import statsmodels.api as sm

X1 = sm.add_constant(X)
logreg = sm.Logit(y,X1).fit()
print(logreg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.523853
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  200
Model:                          Logit   Df Residuals:                      197
Method:                           MLE   Df Model:                            2
Date:                Thu, 14 Feb 2019   Pseudo R-squ.:                  0.2442
Time:                        16:25:30   Log-Likelihood:                -104.77
converged:                       True   LL-Null:                       -138.63
                                        LLR p-value:                 1.974e-15
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9780      0.295     -3.321      0.001      -1.555      -0.401
x1            -0.1344      0.137     -0.980      0.327      -0.403       0.134
x2             1.3981      0.232      6.035      0.000       0.944       1.852
==============================================================================
```

```
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: Interesting Problems

- The loss function expressed as the cross entropy (for $\boldsymbol{y}_i \in \{0, 1\}$) can be re-expressed through the margin $y_i \eta(\boldsymbol{x}_i)$'s (for $\boldsymbol{y}_i \in \{-1, 1\}$) similar to the loss function in the support vector machines.
- The Newton-Raphson method for the GLM is known equivalent to an iteratively reweighted least squares (IRLS) algorithm.
- Subsampled Newton's method for large-scale logistic modeling, as compared to Newton's sketch method.
- Large-scale logistic modeling can be better optimized by first-order method (i.e. SGD algorithm), which can be implemented as a special case of neural network model by Scikit-learn/TensorFlow/Keras/PyTorch

Table of Contents

Softmax Regression

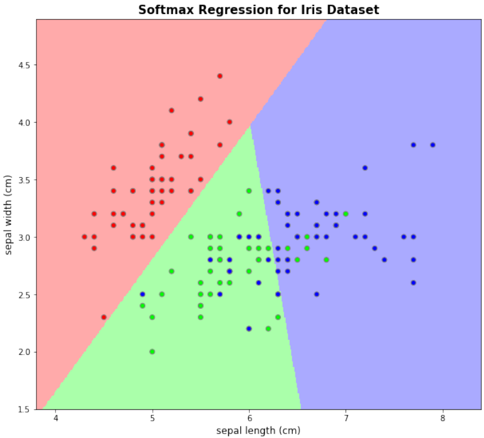
- Softmax regression is also known as "multinomial logistic regression".
- The inverse link function for the probability prediction is given by

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

  Each class has its own dedicated $\boldsymbol{\beta}_k$. By the fact $\sum_{k=1}^K p_k(\boldsymbol{x}) = 1$, we may set the first class as the baseline such that $\boldsymbol{\beta}_1 = \mathbf{0}$.
- The class prediction is given by $\hat{y} = \arg\max_k p_k(\boldsymbol{x})$.
- In Python.Sklearn, use the logistic regression with `multinomial` option:

```
softmaxreg = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=1e10)
softmaxreg.fit(X, y)
```

Softmax Regression: Iris Dataset



See also here for R code demonstration.

Thank You!

Q&A or Email ajzhang@hku.hk.