

VE414 Lecture 22

Jing Liu

UM-SJTU Joint Institute

November 21, 2019

- Recall we have looked at linear models

$$\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \sigma^2\} \sim \text{Normal}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I})$$

where some priors need to be specified in Bayesian context, i.e.

$$\begin{aligned}\boldsymbol{\beta} &\sim \text{Normal}(\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0) \\ \frac{1}{\sigma^2} &\sim \text{Gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0 \sigma_0^2}{2}\right)\end{aligned}$$

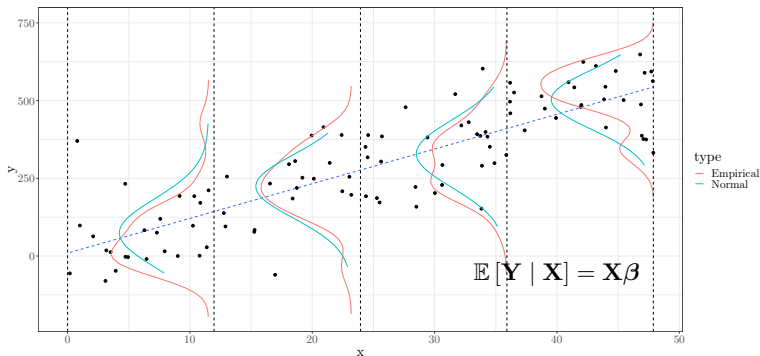
Q: Linear models assume the errors are normally distributed, they are the most widely used models by frequentists, why do frequentists like them so much?

- Without the normal assumption, the followings will not hold,

$$\begin{aligned}\hat{\boldsymbol{\beta}} &\sim \text{Normal}\left(\boldsymbol{\beta}, \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}\right) \\ \hat{\mathbf{e}} &\sim \text{Normal}(\mathbf{0}, \sigma^2 (\mathbf{I} - \mathbf{P}))\end{aligned}$$

without which frequentists will not be able to do inference.

- By comparison, Bayesians are a lot less reliant on the normal assumption,



hence a Bayesian linear model can be as faithful to reality as possible,

$$\mathbf{Y} | \{\mathbf{X}, \boldsymbol{\beta}, \mathbf{Z}\} \sim f_{\mathbf{Y}|\{\mathbf{X}, \boldsymbol{\beta}, \mathbf{Z}\}}$$

$$\boldsymbol{\beta} \sim f_{\boldsymbol{\beta}}$$

$$\mathbf{Z} \sim f_{\mathbf{Z}}$$

- Having the wrong “shape” is often the least worry of the normal assumption.
- Consider the following dataset to see how the normal assumption is invalid

Income	Annual income
Balance	Credit card balance
Default	Whether the card holder has defaulted
Student	Whether the card holder is a student

which is a dataset that the bank want to use to predict credit card default.

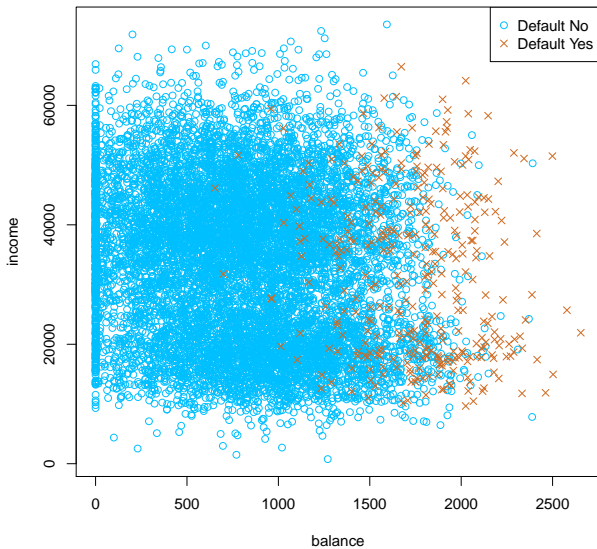
- Notice the response is Default is categorical, i.e. the normal variable

$$\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \sigma^2\} \sim \text{Normal}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I})$$

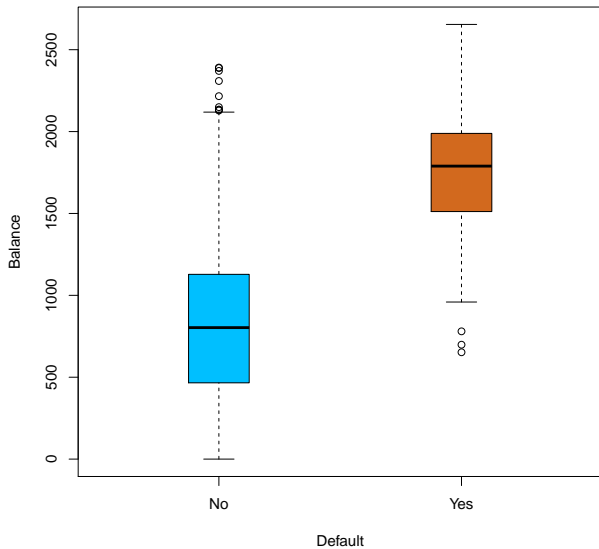
is actually categorical, that is, it can only take a number of discrete values.

- The predictor variables here are categorical or continuous, which are fine.
- So the question here is vary similar to the one in Bayesian decision theory.

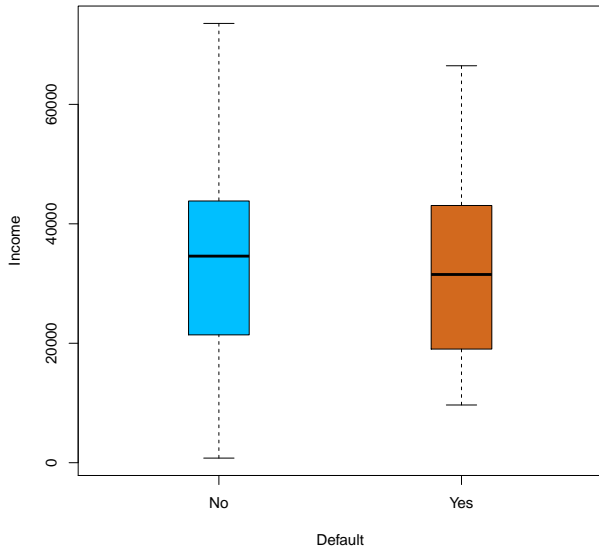
Credit Default



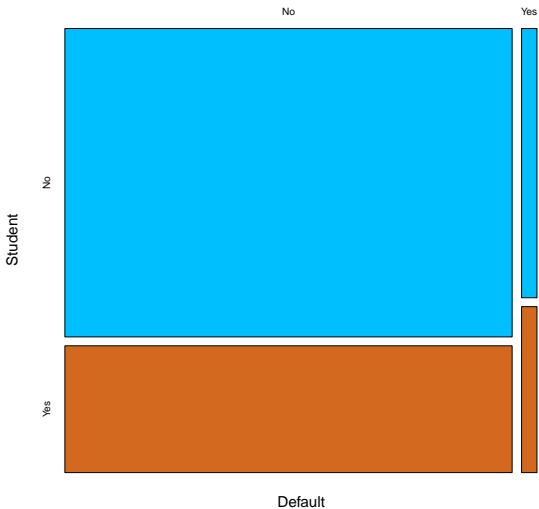
Balance by Default



Income by Default



mosaicplot of Default by Student



Q: Why a linear model is not going to be useful/meaningful here? e.g.

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

where X corresponds to Balance and Y corresponds Default

$$Y = \begin{cases} 0 & \text{if Default} = \text{No}; \\ 1 & \text{if Default} = \text{Yes}. \end{cases}$$

- Recall a simple linear regression models the conditional mean

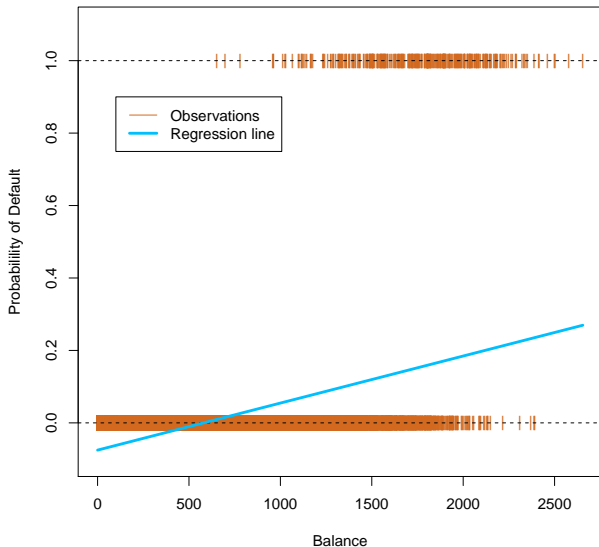
$$\mathbb{E}[Y \mid X = x] = \beta_0 + \beta_1 x$$

by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ for β_0 and β_1 , and we essentially use the estimate

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

to predict the conditional mean for every possible value of $X = x$.

Simple Linear Regression



- To avoid having an estimated probability outside of $[0, 1]$, we model

$$\Pr(Y = 1 \mid X = x)$$

using a function that has the range $[0, 1]$ instead of using

$$\mathbb{E}[Y \mid X = x] = \Pr(Y = 1 \mid X = x) = p(x, \beta) = \beta_0 + \beta_1 x$$

- There are many functions that meet this requirement, if the **logistic** function

$$\Pr(Y = 1 \mid X = x) = p(x, \beta) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

is used, then we will end up with so-called **logistic regression**.

- In general, logistic regression can be thought as modelling the response by

$$Y_i \mid \{\mathbf{X}_i, \beta\} \sim \text{Binomial}(p, 1) \quad \text{where} \quad p = \frac{\exp(\mathbf{x}_i^T \beta)}{1 + \exp(\mathbf{x}_i^T \beta)}$$

- Frequentist would find β_0 and β_1 by maximising likelihood

$$\hat{\beta} = \arg \max_{\mathbf{b}} \{ \mathcal{L}(\mathbf{b}; \mathbf{Y}, \mathbf{X}) \}$$

- Using the maximum likelihood estimate (MLE) of β_0 and β_1 , we have

$$\hat{p}(x, \hat{\beta}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$$

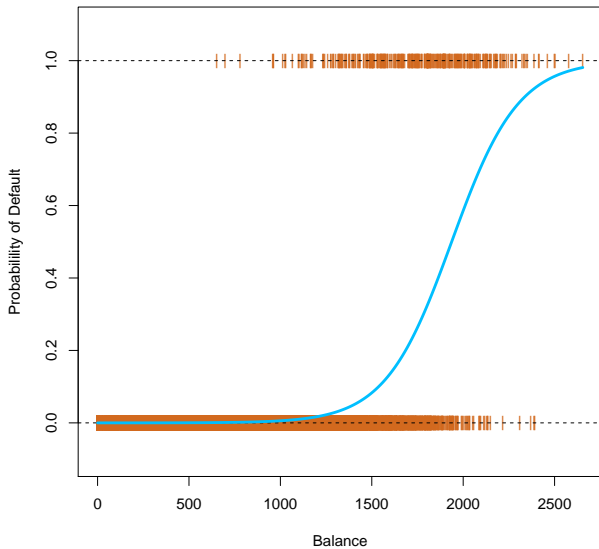
which is an estimate of

$$\Pr(Y = 1 \mid X = x)$$

which gives the likelihood of defaulting in this example, e.g.

$$\Pr(\text{default}=\text{Yes} \mid \text{balance}=1000)$$

Simple logistic Regression



Q: How to obtain MLEs $\hat{\beta}_0$ and $\hat{\beta}_1$? What is the likelihood function here?

$$\mathcal{L}(\beta_0, \beta_1; y_1, \dots, y_n, x_1, \dots, x_n) = \prod_{i=1}^n \Pr(Y = y_i)$$

- Under the assumption of independence and

$$\Pr(Y = 1 \mid X = x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

the negative log-likelihood function is given by

$$\begin{aligned} \ln(\mathcal{L}) &= \ln \left[\prod_{i=1}^n (\Pr(Y = 1 \mid X = x_i))^{y_i} (\Pr(Y = 0 \mid X = x_i))^{1-y_i} \right] \\ &= \sum_{i=1}^n \left(y_i (\beta_0 + \beta_1 x_i) - \ln(1 + \exp(\beta_0 + \beta_1 x_i)) \right) \end{aligned}$$

- We obtain two nonlinear equations when setting the first derivatives to zero

$$\sum_{i=1}^n \left(y_i - \frac{\exp(b_0 + b_1 x_i)}{1 + \exp(b_0 + b_1 x_i)} \right) = 0 \iff \sum_{i=1}^n (y_i - m(\mathbf{b})) = 0$$

$$\sum_{i=1}^n x_i \left(y_i - \frac{\exp(b_0 + b_1 x_i)}{1 + \exp(b_0 + b_1 x_i)} \right) = 0 \iff \sum_{i=1}^n x_i (y_i - m(\mathbf{b})) = 0$$

- We can then solve the two nonlinear equations numerically, for example,

```
> credit.LG = glm(default~balance, family = binomial,
+               data = credit.df)
>
> coef(credit.LG)
```

(Intercept)	balance
-10.651330614	0.005498917

R has a build-in logistic regression routine using Frequentist's approach.

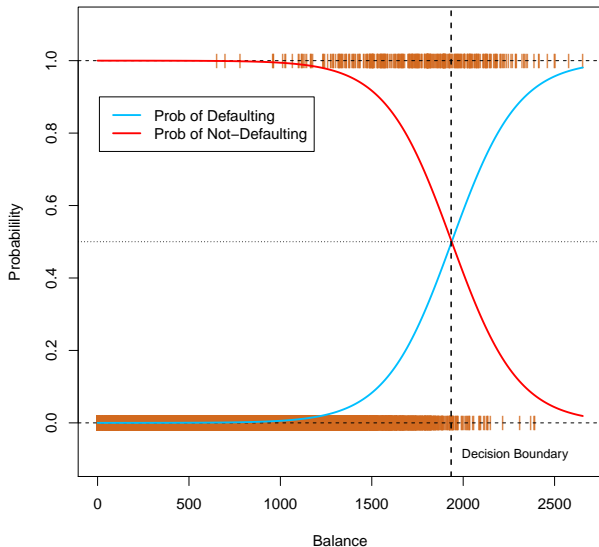
- Thus, for the individual with Balance = 1000, our prediction is

$$\begin{aligned}\hat{p} = \hat{\text{Pr}}(Y = 1 \mid X = 1000) &= \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)} \\ &= \frac{\exp(-10.6513 + 0.0055 \cdot 1000)}{1 + \exp(-10.6513 + 0.0055 \cdot 1000)} \\ &= 0.00575\end{aligned}$$

```
> newdata.df = data.frame(balance = 1000)
>
> phat = predict(credit.LG, newdata = newdata.df,
+               type = "response")
> phat
```

```
1
0.005752145
```


Simple logistic Regression



- In terms of decision theorem, the following can be used as a classifier

$$z = \arg \max_{z \in \{0,1\}} \{z \cdot \hat{p} + (1 - z) \cdot (1 - \hat{p})\}$$

which classifies the one with Balance=1000 into the “not defaulting” class.

Q: How would a Bayesian approach the same problem?

$$Y \mid \{\beta_0, \beta_1, X\} \sim \text{Binomial}(p, 1)$$

where

$$p = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

Q: Given n independent (x_i, y_i) , what is the posterior if we use a normal prior?

$$\boldsymbol{\beta} \sim \text{Normal}(\boldsymbol{\mu}, \text{diag}(\sigma_0^2, \sigma_1^2))$$

- In this case, the posterior takes the following form

$$f_{\beta|\{\mathbf{Y}, \mathbf{X}\}} \propto \mathcal{L}f_{\beta} = \prod_{i=1}^n \left(\frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)} \right)^{y_i} \left(\frac{1}{1 + \exp(\beta_0 + \beta_1 x_i)} \right)^{1-y_i} \\ \cdot \exp \left(-\frac{1}{2\sigma_0^2} (\beta_0 - \mu_0)^2 - \frac{1}{2\sigma_1^2} (\beta_1 - \mu_1)^2 \right)$$

- The normalisation constant has no analytic form, but MAP can be used here

$$\begin{aligned} \hat{\beta}_{\text{MAP}} &= \arg \max_{\mathbf{b}} \{ \mathcal{L}(\mathbf{b}; \mathbf{Y}, \mathbf{X}) \cdot f_{\beta}(\mathbf{b}) \} \\ &= \arg \max_{\mathbf{b}} \{ \ln(\mathcal{L}) + \ln(f_{\beta}) \} \\ &= \arg \max_{\mathbf{b}} \left\{ \sum_{i=1}^n \left(y_i (b_0 + b_1 x_i) - \ln(1 + \exp(b_0 + b_1 x_i)) \right) \right. \\ &\quad \left. - \frac{1}{2\sigma_0^2} (b_0 - \mu_0)^2 - \frac{1}{2\sigma_1^2} (b_1 - \mu_1)^2 \right\} \\ &= \hat{\beta}_{\text{MLE}} \quad \text{as} \quad \sigma_0^2, \sigma_1^2 \rightarrow \infty \end{aligned}$$

- Recall frequentist uses the following as the prediction

$$\hat{p} = \frac{\exp\left(\mathbf{x}^T \hat{\boldsymbol{\beta}}_{\text{MLE}}\right)}{1 + \exp\left(\mathbf{x}^T \hat{\boldsymbol{\beta}}_{\text{MLE}}\right)} \quad \text{where} \quad \mathbf{x} = \begin{bmatrix} 1 \\ 1000 \end{bmatrix}$$

and the following could be used as a classifier

$$z = \arg \max_{z \in \{0,1\}} \{z \cdot \hat{p} + (1 - z) \cdot (1 - \hat{p})\}$$

Q: What is the MAP classifier in this case? How to obtain the MAP classifier?

$$\begin{aligned} z &= \arg \max_{z \in \{0,1\}} f_{Y^*|\{\mathbf{Y}, \mathbf{X}, X^*\}}(z \mid \mathbf{y}, \mathbf{X}, 1000) \\ &= \arg \max_{z \in \{0,1\}} \int_{\mathcal{D}} f_{\{Y^*, \boldsymbol{\beta}\}|\{\mathbf{y}, \mathbf{X}, X^*\}}(z, \boldsymbol{\beta} \mid \mathbf{y}, \mathbf{X}, 1000) d\boldsymbol{\beta} \\ &= \arg \max_{z \in \{0,1\}} \int_{\mathcal{D}} f_{Y^*|\{X^*, \boldsymbol{\beta}\}}(z \mid 1000, \boldsymbol{\beta}) f_{\boldsymbol{\beta}|\{\mathbf{Y}, \mathbf{X}\}}(\boldsymbol{\beta} \mid \mathbf{y}, \mathbf{X}) d\boldsymbol{\beta} \end{aligned}$$

- Thus it becomes a Bayesian computational problem again.
- Stan is a tool for building *standard* Bayesian model using *standard* MCMC.
- The best way to use Stan is via R/Python,

```
> # install.packages("rstan")  
> library(rstan)
```

- A Stan program usually has three components:

```
data {  
  int<lower=0> N;  
  vector[N] x;  
  int<lower=0,upper=1> y[N];  
}  
parameters {  
  real beta0;  
  real beta1;  
}  
model {  
  y ~ bernoulli_logit(beta0 + beta1 * x);  
}
```

- Once the Stan program on the last page is saved, it can be run within R

```
> fit = stan("logistic_credit_simple_414.stan",
+           data = list(
+               x = credit.df$balance,
+               y = credit.df$default,
+               N = nrow(credit.df)),
+           chains = 4,
+           cores = 4,
+           iter = 1000,
+           warmup = 200))

> beta_simple_summary =
+   summary(fit, pars = c("beta0", "beta1"),
+           probs = c(0.025, 0.9725))$summary
> beta_simple_summary[, c("mean", "2.5%", "97.25%")]
```

	mean	2.5%	97.5%
beta0	-9.241026120	-1.128547e+01	-2.623256862
beta1	0.004594058	-1.155165e-05	0.005876377

- There are some difference even non-informative priors are used by default

```
> confint(credit.LG)
```

	2.5 %	97.5 %
(Intercept)	-11.383288936	-9.966565064
balance	0.005078926	0.005943365

```
> beta_simple_summary[, c("2.5%", "97.5%")]
```

	2.5%	97.5%
beta0	-1.128547e+01	-2.623256862
beta1	-1.155165e-05	0.005876377

- An informative prior can be specified in the model section

```
model {
  beta0 ~ normal(0, 1);           #New
  beta1 ~ normal(0, 0.1);        #New
  y ~ bernoulli_logit(beta0 + beta1 * x);
}
```

- Run this new stan program,

```
> fit_norm = stan("logistic_credit_norm_414.stan",  
+               data = list(  
+                   x = credit.df$balance,  
+                   y = credit.df$default,  
+                   N = nrow(credit.df)),  
+               chains = 4, cores = 4,  
+               iter = 1000, warmup = 200))  
>  
> beta_norm_summary =  
+   summary(fit_norm, pars = c("beta0", "beta1"),  
+         probs = c(0.025, 0.9725))$summary  
>  
> beta_norm_summary[, c("2.5%", "97.5%")]
```

	2.5%	97.5%
beta0	-1.046182e+01	-2.61198411
beta1	1.808603e-05	0.00539212

- The actual sample from stan can be extracted for further analysis, e.g

```
> beta0 = extract(object = fit_normal ,  
+                 pars = "beta0")  
>  
> head(beta0$beta0)
```

```
[1] -8.289337 -4.948232 -8.790315  
[4] -9.277918 -9.023087 -4.939379
```

```
> getmode = function(v) {  
+   uniqv = unique(v)  
+   uniqv[which.max(tabulate(match(v, uniqv)))]  
+ }  
>  
> getmode(beta0$beta0)
```

```
[1] -9.593992
```

- For very common models, the following package is useful,

```
> library(rstanarm)

> credit.bayesian.LG = stan_glm(default~balance,
+                               family = binomial(link = "logit"),
+                               prior_intercept = normal(0, 1),
+                               prior = normal(0,0.1), chains = 4,
+                               iter = 1000, data = credit.df)

> posterior_interval(credit.bayesian.LG, prob=0.95)
```

	2.5%	97.5%
(Intercept)	-7.431983	-6.769537753
balance	0.002993	0.003419604

```
> confint(credit.LG)
```

	2.5 %	97.5 %
(Intercept)	-11.383288936	-9.966565064
balance	0.005078926	0.005943365

- The posterior predictive distribution can be obtained in the following way

```
> newdata.df = data.frame(balance = 1000)
```

- Take a sample from the posterior predictive distribution

```
> post_pred_sample = posterior_predict(draws = 2000,  
+   credit.bayesian.LG, newdata = newdata.df)  
>  
> post_pred_sample[1:20,]
```

```
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

- The estimate is simply the sample mean, which differs slightly from MLE

```
> mean(post_pred_sample); phat
```

```
[1] 0.0155  
[1] 0.005752145
```

- It also classifies the one with balance=1000 into the “not Defaulting” class.