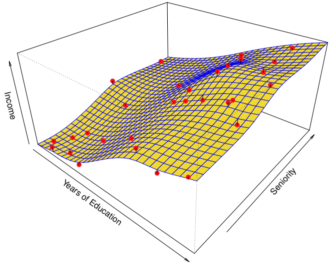
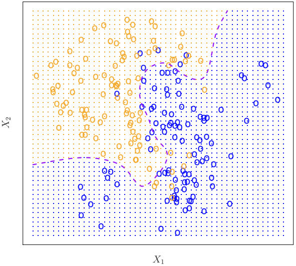
Classification

Classification

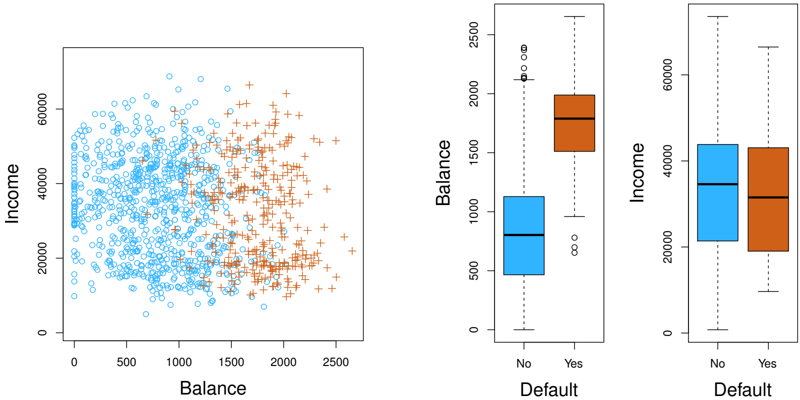
- In many applications, the response is not a quantitative value and instead represents a class, e.g., $y \in$ {student, non-student}, $y \in$ {while, yellow, green}
- Yet based on the observation of some features, we would like to predict the class (what we refer to as the classification)
- Regression vs classification





Classification

Example. Predicting default cases on the credit card (unable to pay the credit card), based on the income and current balance



(one immediate observation is probably balance is a more useful feature)

Binary Classification

- In simple regression for a single feature x we fitted a line $y = \beta_0 + \beta_1 x$ to the data
- In binary classification with only one feature, we don't have values any more, but two classes (say class 0 and class 1)
- Can we do the fit in a way that the sign of $\beta_0 + \beta_1 x$ becomes an indicator of the class for us?
- In other words, for a given feature x_t, we make a decision based on the following:

$$y_t = \begin{cases} 1 & \beta_0 + \beta_1 x_t > 0 \\ 0 & \beta_0 + \beta_1 x_t < 0 \end{cases},$$

- A smooth function (called Sigmoid – also inverse Logit) that takes almost binary values $0, 1$ based on the sign of the input z is

Sigmoid for continuous

$$\frac{e^z}{1 + e^z} \approx \begin{cases} 1 & z >> 0 \\ 0 & z << 0 \end{cases}$$

Simple Example

input (feature) : $x \in \mathbb{R}$

output (response) : $y \in \{0, 1\}$

$$y = \text{Sign}(\beta_0 + \beta_1 x)$$

problem :

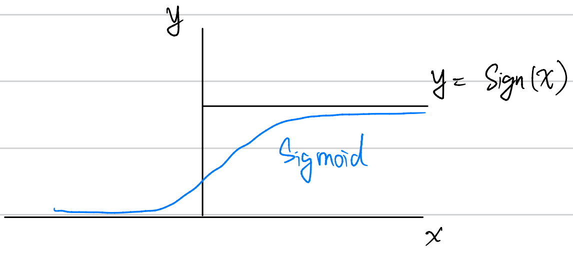
Sign Function is not continuous

Regression : $y = \beta_0 + \beta_1 x$

$(x_1, y_1) \dots (x_n, y_n)$ fit β_0, β_1

such that $y_i \simeq \text{Sign}(\beta_0 + \beta_1 x_i)$, $i = 1, \dots, n$

minimize $\sum_{i=1}^{n} (y_i - \text{Sign}(\beta_0 + \beta_1 x_i))^2$



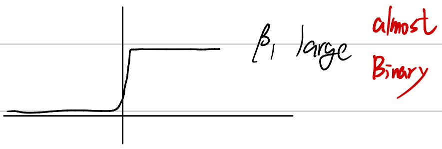
Sigmoid $\text{Sign}(\beta_0 + \beta_1 x) \rightarrow \sigma(\beta_0 + \beta_1 x)$

$$P(y=1|x) = \sigma(\beta_0 + \beta_1 x)$$

$$\rightarrow P(y=0|x) = 1 - \sigma(\beta_0 + \beta_1 x)$$

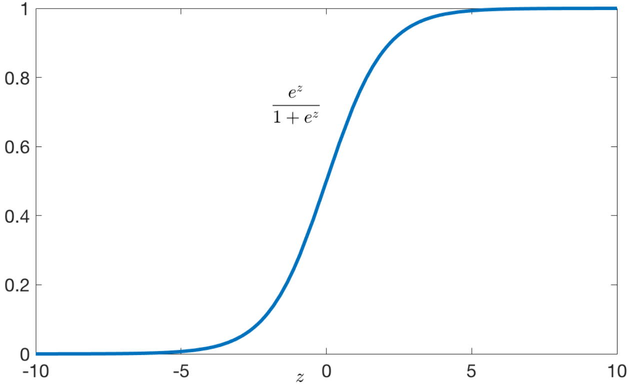
β_0 : offset

β_1 : slope

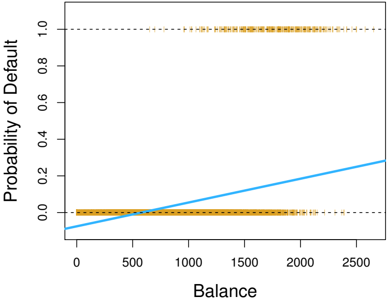


Binary Classification

– When we have a smooth approximation of the sign function, learning the parameters β_0 and β_1 is numerically easier



Binary Classification



Trying to treat the classification problem as a regression problem does not produce reasonable results!

Steps to solve problem

Regression

$f_\theta(y|x)$

$-\log f(y|x)$

Max Likelihood

of find $f_\theta(y|x)$

Classification

label

$P_\theta(Y=\ell|x)$

$-\log P(Y=\ell|x)$

Max Likelihood

How Does Binary Classification Work?

- We somehow learn β_0 and β_1 from the training data (will be explained soon)
- We are given a test point x_t, for which we evaluate $\beta_0 + \beta_1 x_t$
- We pass this quantity to our smooth sign approximation

$$p(x_t) = \frac{e^{\beta_0 + \beta_1 x_t}}{1 + e^{\beta_0 + \beta_1 x_t}}$$

- If $p(x_t)$ was closer to 1 our prediction of the class for x_t is class one (e.g., $p(x_t) = 0.7$) and if $p(x_t)$ was closer to 0 our prediction of the class for x_t is class zero (e.g., $p(x_t) = 0.3$)
- Now that $p(\cdot)$ generates some value between zero and one for us, one immediate interpretation for it is being the probability of label 1

$$p(x_t) = \mathbb{P}(y = 1 | x_t) = 1 - \mathbb{P}(y = 0 | x_t)$$

  so if $p(x_t) = 0.7$, then the test label is 1 with probability 0.7, and 0 with probability 0.3

How to Do the Training for the Simple Logistic Regression?

- Many of the classification techniques you see in this course only differ in the way that we model

$$\mathbb{P}(Y = \ell | x_t)$$

- We observe samples $(x_1, y_1), \cdots (x_n, y_n)$, where $y_i \in \{0, 1\}$
- We want to determine β_0 and β_1 such that the probability of assigning the right labels is maximized

$$\arg\max_{\beta_0, \beta_1} \; \mathbb{P}(Y_1 = y_1, \cdots Y_n = y_n | X_1 = x_1, \cdots, X_n = x_n, \beta_0, \beta_1)$$

- Basically, we want to find the ML estimates for β_0 and β_1

Model : $P(Y=1|x) = \sigma(\beta_0+\beta_1 x) = \dfrac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$

$P(Y=0|x) = 1-\sigma(\beta_0+\beta_1 x) = 1-\sigma(\beta_0+\beta_1 x)$

Training Data : $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$
(Samples)

To derive ML

$\rightarrow P(Y_1=y_1, Y_2=y_2, \dots, Y_n=y_n \mid x_1, \dots x_n, \beta_0, \beta_1)$

Since independent $= \prod_{i=1}^{n} P(Y_i=y_i \mid x_i, \beta_0, \beta_1)$

only estimates β_0, β_1
not x_i

– Since our samples are independent, we get

$$\begin{aligned}\mathbb{P}(Y_1 = y_1, \cdots Y_n = y_n | x_1, \cdots, x_n, \beta_0, \beta_1) &= \prod_{i=1}^{n} \mathbb{P}(Y_i = y_i | x_i, \beta_0, \beta_1) \\ &= \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)) \\ &= \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}\end{aligned}$$

where the first equality is thanks to

$$p(x_i) = \mathbb{P}(Y = 1 | x_i) = 1 - \mathbb{P}(y = 0 | x_i)$$

– So we ultimately want to find β_0 and β_1 that maximize

$$\prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} = \prod_{i=1}^{n} \left(\frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \right)^{y_i} \left(1 - \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \right)^{1-y_i}$$

Some Notes on The Logistic Regression

−log( ) → convert Max to min , then we can use GD

- In logistic regression, we end up with a more complex cost function to optimize (after applying the negative log we get)

$$L(\beta_0, \beta_1) = -\log\left(\prod_{i=1}^{n}\left(\frac{1}{1+e^{-\beta_0-\beta_1 x_i}}\right)^{y_i}\left(\frac{1}{1+e^{\beta_0+\beta_1 x_i}}\right)^{1-y_i}\right)$$

$$= -\sum_{i=1}^{n} -y_i \log\left(1+e^{-\beta_0-\beta_1 x_i}\right) - (1-y_i)\log\left(1+e^{\beta_0+\beta_1 x_i}\right)$$

$$= \sum_{i=1}^{n} y_i \log\left(1+e^{-\beta_0-\beta_1 x_i}\right) + (1-y_i)\log\left(1+e^{\beta_0+\beta_1 x_i}\right)$$

- This function is convex and can be nicely minimized using gradient descent. You may see examples in the homework!

Turns out that

$$f(z) = y \cdot \log(1+e^{-z}) + (1-y)\log(1+e^{z})$$ is convex in z

$$z = \beta_0 + \beta_1 x$$

$$f(\beta_0, \beta_1) = y \cdot \log(1+e^{-\beta_0-\beta_1 x}) + (1-y)\log(1+e^{\beta_0+\beta_1 x})$$ is convex in β_0, β_1

Logistic Regression has no closed form solution to β_0 and β_1

What Happens for More than One Feature?

- In case of multiple features, only minor modification is required
- We still try to maximize $\prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$, but now we have

$$p(x_t) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}$$

- We run the maximization to estimate $\beta_0, \beta_1, \cdots, \beta_p$
- In practice you never have to do the maximization and most software such as R, Python and Matlab have packages to do that numerically

① Only one feature x, $y \in \{0, 1\}$ $\Rightarrow$ ML formula is on P.32

② Multiple features, $x_1, x_2, \dots x_n$, $y \in \{0, 1\}$

$$P(Y=1|X) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)$$

What Happens for More than Two Classes?

- Example, based on some features such as city, year of education and number of publications, classify the students of a class into undergrads, Masters, and PhDs
- Recall our method of classification in the binary case, we evaluated $p(x_t)$ which was technically $\mathbb{P}(Y = 1|x_t)$ and if it was closer to 1 then our class prediction was 1, if it was small, then $\mathbb{P}(Y = 0|x_t) = 1 - \mathbb{P}(Y = 1|x_t)$ would be large and our prediction is class zero
- One way of interpreting this is evaluating $\mathbb{P}(Y = k|x_t)$ for $k = 0, 1$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label
- Now for K labels, we evaluate $\mathbb{P}(Y = k|x_t)$ for $k = 1, 2, \cdots, K$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label

What Happens for More than Two Classes?

- For K labels, we evaluate $\mathbb{P}(Y = k|x_t)$ for $k = 1, 2, \cdots, K$ and the k that produces the largest value for $\mathbb{P}(Y = k|x_t)$ is our predicted label
- When we have $K > 2$ labels (e.g., $y \in$\{while, yellow, green\}) and p features $x_1, x_2, \cdots, x_p$, we fit K models parametrized by

$$\begin{aligned}
\text{Label 1:} &\quad \{\beta_0^{(1)}, \beta_1^{(1)}, \cdots, \beta_p^{(1)}\} \\
\text{Label 2:} &\quad \{\beta_0^{(2)}, \beta_1^{(2)}, \cdots, \beta_p^{(2)}\} \\
&\vdots \\
\text{Label K:} &\quad \{\beta_0^{(K)}, \beta_1^{(K)}, \cdots, \beta_p^{(K)}\}
\end{aligned}$$

- For this problem we consider the following form:

$$p_k(\boldsymbol{x}) = \mathbb{P}(Y = k|\boldsymbol{x}) = \frac{e^{\beta_0^{(k)} + \cdots + \beta_p^{(k)} x_p}}{e^{\beta_0^{(1)} + \cdots + \beta_p^{(1)} x_p} + \cdots + e^{\beta_0^{(K)} + \cdots + \beta_p^{(K)} x_p}}$$

- What is the sum of all $\mathbb{P}(Y = k|\boldsymbol{x})$ for a fixed $\boldsymbol{x}$?

What if 3 classes?

Classes $y \in \{0, 1, 2\}$

$\rightarrow P(Y=0|X) + P(Y=1|X) + P(Y=2|X) = 1$

Therefore, for more two classes:

$P(Y=l|X)$ $\quad l \in$ the class labels

Have to deal with each model individually

$$\begin{cases} P(Y=1|X): \beta_0^{(1)}, \beta_1^{(1)}, \dots \beta_p^{(1)} \\ P(Y=2|X): \beta_0^{(2)}, \beta_1^{(2)}, \dots \beta_p^{(2)} \\ \vdots \\ P(Y=L|X): \beta_0^{(L)}, \beta_1^{(L)}, \dots \beta_p^{(L)} \end{cases}$$

Then Make Sure

$$\sum_{Y=1}^{L} P(Y=l|X) = 1$$

Some R Simulations

- Let's preform some basic classification tasks in R!