PSTAT 126

Regression Analysis

Laura Baracaldo

Lecture 12

Logistic Regression & Neural Networks

Binomial Regression Model

Suppose that for each subject we observe a binary response variable $y \in \{0,1\}$ and a vector of predictors $x = (x_1, \ldots, x_p)^T$. We assume the response has a bernoulli distribution:

$$P(y_i) = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{if } y_i = 0 \end{cases}$$

We further assume that the Y_i are independent. For each individual $i = 1, \ldots, n$, we want to model the probabilities:

$$P(y_i = 1 | x_{i1}, \ldots, x_{ip}) = f(x_1, \ldots, x_p)$$
$$P(y_i = 0 | x_{i1}, \ldots, x_{ip}) = 1 - P(y_i = 1 | x_{i1}, \ldots, x_{ip})$$

Binomial Regression Model

Why can not we just model f as a linear combination of the predictors?

- Because $f(x_1, \ldots, x_p) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p \in \mathbb{R}$ and we want to restrict the co domain/image of the function $f(x_1, \ldots, x_p) \in [0, 1]$.

How to properly link the probability of a binary response to a linear combination of real predictors?

$$p_i = P(y_i = 1 | x_{i1}, \ldots, x_{ip}) = g(\beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi})$$

Such that $g : \mathbb{R} \to [0, 1]$.

Link Function g

There are several possibilities for this function g:

- Logit (Canonical link function): $g(x) = \frac{e^x}{1+e^x}$
- Probit: $g(x) = \Phi(x)$, where Φ is the standard normal cumulative function.
- Complementary log-log: $g(x) = 1 - e^{-e^x}$

Logistic Regression

We consider the binary response variable $Y_i|\boldsymbol{x}_i \overset{\text{ind}}{\sim} Ber(p_i)$ for observations $i = 1, \ldots, n$, with $p_i = Pr(Y_i = 1)$. The goal is to model this probability based on a set of predictors $x_{i1}, \ldots, x_{ip}$. The logistic regression describes the relationship between p_i and the predictors as:

$$p_i = E(Y_i|\boldsymbol{x}_i) = \frac{\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i^T\boldsymbol{\beta})} \in [0, 1]$$

Or equivalently:

$$\boldsymbol{x}_i^T\boldsymbol{\beta} = logit(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) \in (-\infty, \infty)$$

Where $\boldsymbol{x}_i = (1, x_{i1}, \ldots, x_{ip})^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$.

Odds Ratio

Odds are sometimes a better scale than probability to represent chance. The odds of an event are the probability that the event occurs divided by the probability that the event does not occur. In logistic regression it can be expressed as:

$$o_i = \frac{p_i}{1 - p_i} = \exp(\boldsymbol{x}_i^T \boldsymbol{\beta}) \in [0, \infty)$$

- One mathematical advantage of odds is they are unbounded above which makes them more convenient for some modeling purposes.
- We use the odds ratio to understand the effect of a predictor.

Interpreting Odds

Continuous predictors:Let's consider the binary model with p predictors, such that the odds can be expressed: as:

$$o = \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)$$

For predictor x_j we can interpret β_j in terms of Odds: Holding fixed the rest of predictors, a unit increase in x_j will increase the odds by a factor of $\exp(\beta_j)$: $\exp(\beta_j) = o_{x_j+1}/o_{x_j}$. This implies:

- If $\beta_j > 0$, $p = P(Y = 1)$ will increase as x_j increases (when fixing the other predictors).
- If $\beta_j < 0$ the odds will decrease as x_j increases (when fixing the other predictors).

Interpreting Odds

Categorical predictors: The odds ratio compares the odds of the event occurring at different levels of the predictor. For a two-level factor with dummy variable:

$$z = \begin{cases} 1 & \text{if Level A} \\ 0 & \text{if Level B} \end{cases}$$

The odds of a model with p predictors and one two-level factor is expressed as:

$$o = \exp(\beta_0 + \beta_A z + \beta_1 x_1 + \ldots + \beta_p x_p)$$

When holding fixed all predictors, the odds of *group A* will increase with respect to *group B* by a factor of $\exp(\beta_A)$: $\exp(\beta_A) = o_A/o_B$ This implies:

- If $\beta_A > 0$ the odds will be larger for group A compared to the odds for group B.
- If $\beta_A < 0$ the odds will be smaller for group A compared to the odds for group B.

Maximum Likelihood Estimation

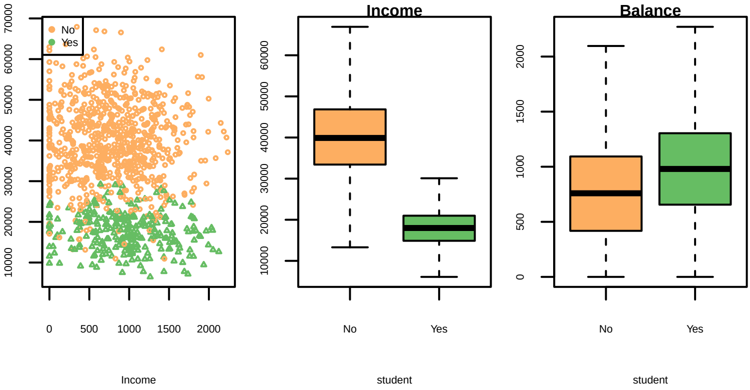
The log-likelihood can be written as:

$$
\begin{aligned}
l = \log(L) &= \log\left(\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}\right) \\
&= \sum_{i=1}^{n}\left\{y_i \boldsymbol{x}_i^T\boldsymbol{\beta} - \log\left(1+\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})\right)\right\}
\end{aligned}
$$

Maximization can not be performed analytically, so we must use numerical optimization. We can apply algorithms such as Newton-Raphson method with Fisher Scoring.

Data Example - Credit Card Payment

Let us consider the response variable: Y : Whether an individual will default on their credit card. x_1: Annual income, x_2: Monthly Credit card balance, x_3: Is the individual a student or not.



Data Example - MLE

```r
## glm function for Binomial response:
#It runs a Newton Raphson method with Fisher Scoring
lmod<- glm(default~student+balance+income, data=Default, family=binomial)
summary(lmod)
```

```
##
## Call:
## glm(formula = default ~ student + balance + income, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

Data Example - Probability Prediction

```
x0<- data.frame(balance=1500, income=40000, student="No")
predict(lmod, x0, type="response")
```

```
##         1
## 0.1049919
```

We can predict the class (default=TRUE or default=FALSE), by setting a threshold a for the predicted probability:

$$\hat{y} = \begin{cases} 1 & \text{if } p \geq a \\ 0 & \text{if } p < a \end{cases}$$

We could set for example $a = 0.5$. Or, if the credit card company is super conservative in predicting individuals who are at risk for default, one might set $a = 0.1$.

Classification Performance

For a binary classifier, use confusion matrix:

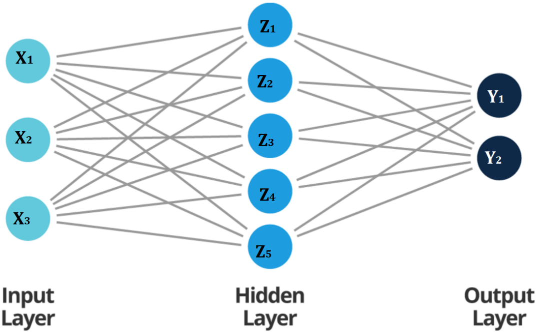
	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	True Negative (**TN**)	False Positive (**FP**)
$y = 1$	False Negative (**FN**)	True Positive (**TP**)

- **True positive rate (TPR)**=TP/(TP+FN)
- **False positive rate (FPR)**=FP/(FP+TN)

Neural Networks

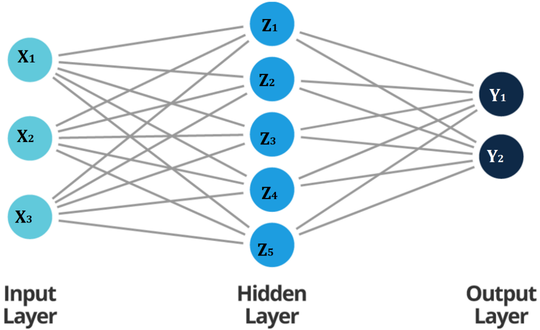
- NN is algorithms are inspired by the human brain to performs a particular task or functions.
- The neural network is a set of connected input/output units in which each connection has a weight associated with it.
- In the learning phase, the network learns by adjusting the weights to predict the correct class label of the given inputs.

Basic Neural Network



- Depth of NN = Number of hidden layers (**Deep Learning**)
- Nowadays, deep networks have tens or even hundreds of hidden layers.

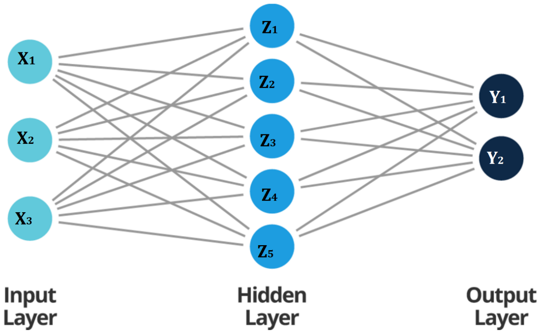
Basic Neural Network



- Hidden units Z's: derived features as functions of linear combinations of the inputs:

$$Z_m = \sigma(\alpha_{0m} + \alpha_{1m}X_1 + \alpha_{2m}X_2 + \alpha_{3m}X_3)$$

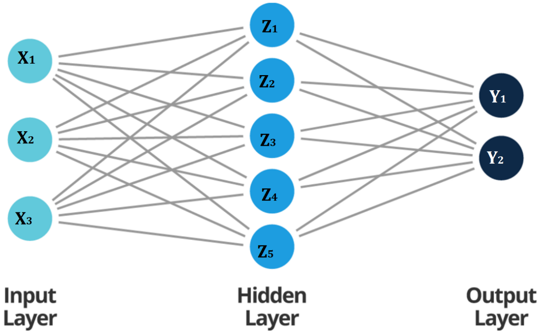
Basic Neural Network



- Output Y's: functions of linear combinations of the hidden units.

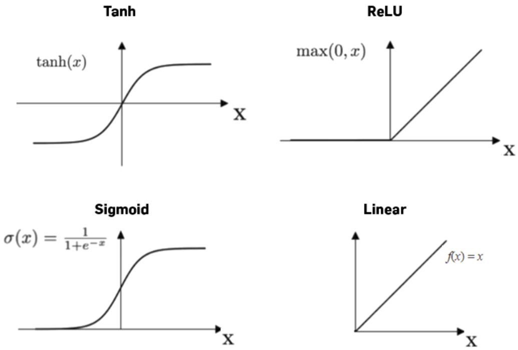
$$Y_k = g(\beta_{0k} + \beta_{1k} Z_1 + \beta_{2k} Z_2 + \beta_{3k} Z_3 + \beta_{4k} Z_4 + \beta_{5k} Z_5)$$

Basic Neural Network



- To train a neural network is essentially equivalent to estimate all α's and all β's

Activation Function



- If we use identity activation function, the entire NN model collapses to a linear model

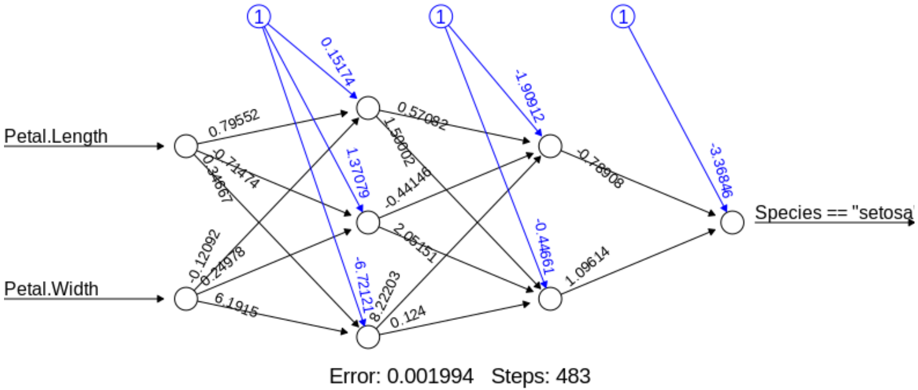
Neural Netwroks - Data Example

```r
# install package

str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
sample1<- sample( dim(iris)[1], 130)
iristrain<- iris[sample1,]
iristest<- iris[-sample1,]
require(neuralnet)

# fit neural network
softplus <- function(x) log(1 + exp(x))
nn <- neuralnet((Species == "setosa") ~ Petal.Length + Petal.Width, iristrain,
                linear.output = FALSE, hidden = c(3, 2), act.fct = softplus)
```

Neural Netwroks - Data Example



Neural Netwroks - Data Example

```
Predict<-compute(nn,iristest)
prob <- Predict$net.result;pred <-ifelse(prob>0.5, 1, 0)
datf<-data.frame(prediction=pred, actual=iristest$Species);datf
```

```
##     prediction     actual
## 10           1     setosa
## 17           1     setosa
## 20           1     setosa
## 42           1     setosa
## 55           0 versicolor
## 58           0 versicolor
## 62           0 versicolor
## 65           0 versicolor
## 80           0 versicolor
## 83           0 versicolor
## 87           0 versicolor
## 99           0 versicolor
## 106          0  virginica
## 110          0  virginica
## 111          0  virginica
## 122          0  virginica
## 124          0  virginica
## 127          0  virginica
## 131          0  virginica
## 144          0  virginica
```