Regression versus Classification

Regression versus Classification

- Variables can be characterized as either quantitative or qualitative.
- Examples of quantitative variables include a person's age, height, or income, the value of a house, and the price of a stock. Quantitative variables take on numerical values.
- In contrast, qualitative variables take on values in one of K different classes, or categories.
- Examples of qualitative variables include a person's gender (male or female), the brand of product purchased (brand A, B, or C),or whether a person defaults on a debt (yes or no).
- When the response variable Y is quantitative, these machine learning problems are referred to as regression problems.
- When the response variable Y is qualitative, these machine learning problems are referred to as classification problems.

Application of Machine Learning Classification Tasks in Real Life

Classification problems occur often, perhaps even more so than regression problems. Some examples include:

- A person arrives at a clinic with a set of symptoms and the doctor has to decide which of three medical conditions the patient is suffering from.
- An online financial platform must decide whether or not a transaction being performed on the site is fraudulent based on past transaction history, and other information such as IP address.
- A biologist has to figure out which DNA mutations are disease causing and which are not.

Loss function for the Classification Setting

- We saw how the quadratic loss function was the natural loss function to consider in regression settings:

$$E_{f(X,Y)}(Y - f(X))^2 = E(L(Y, f(X)))$$

where $L(a, \delta) = (a - \delta)^2$ is the loss function with unknown state a and decision δ.

- But many concepts we discussed earlier, such as the bias-variance trade-off, carry over to the classification setting.
- But there are some modifications due to the fact that Y is no longer numerical.

- In the case of classification we consider the $0 - 1$ loss function:

$$\begin{aligned} L(a, \delta) &= 0 \qquad \text{if } a = \delta \text{ and} \\ &= 1 \qquad \text{if } a \neq \delta. \end{aligned}$$

i.e., $L(a, \delta) = I(a \neq \delta)$ where $I(A)$ is the indicator function of set A.

Best classifier formulation and derivation

- Suppose we seek to estimate the best classifier $f(X)$ based on the $0-1$ loss function.
- The formulation of the best classifier problem is similar to the case of identifying $E(Y|X)$ as the best regressor previously.
- Assume Y can take two labels a and b, and the joint probability distribution of (X, Y) is $\pi(x, y)$.
- The expected loss when $(X, Y) \sim \pi(x, y)$ is

$$
\begin{aligned}
E_{\pi(x,y)} L(Y, f(X)) &= E_{\pi(x)} E_{\pi(y|x)} L(Y, f(X)) \\
&= E_{\pi(x)} \Big[\underbrace{L(a, f(X))\pi(a|X) + L(b, f(X))\pi(b|X)}_{(*)} \Big]
\end{aligned}
$$

Best classifier formulation and derivation (cont.)

- To minimize $E_{\pi(x,y)} L(Y, f(X))$ with respect to f, we minimize the expression $(*)$ inside the square brackets for each X. Thus, if

$$\pi(a|X) < \pi(b|X)$$

it is best to take $f(X) = b$ since in that case,

$$(*) = 1 \cdot \pi(a|X) + 0 \cdot \pi(b|X) = \pi(a|X),$$

the smaller of $\pi(a|X)$ and $\pi(b|X)$.

- Conversely, if

$$\pi(a|X) \geq \pi(b|X)$$

it is best to take $f(X) = a$ since in that case,

$$(*) = 0 \cdot \pi(a|X) + 1 \cdot \pi(b|X) = \pi(b|X),$$

which is again the smaller of $\pi(a|X)$ and $\pi(b|X)$.

Best classifier formulation and derivation (cont.)

- Putting this all together, the best classfier $f(X)$ is the one that predicts the label of y as follows:

$$
\begin{aligned}
f(X) &= a && \text{if } \pi(a|X) > \pi(b|X), \\
&= b && \text{if } \pi(a|X) < \pi(b|X), \text{ and} \\
&= a \text{ or } b, && \text{if } \pi(a|X) = \pi(b|X)
\end{aligned}
$$

- Since $\pi(a|X) + \pi(b|X) = 1$, the three conditions above are equivalent to $\pi(a|X) > 0.5$, $\pi(a|X) < 0.5$ and $\pi(a|X) = 0.5$.
- However, since $\pi(x, y)$ is unknown, the conditional probabilities $\pi(a|X)$ and $\pi(b|X)$ are unknown and have to be estimated based on a pre-specified class $\mathcal{C}$.

Learning a classifier from training samples

- We seek to estimate $\hat{f}$ from class $\mathcal{C}$ based on a training dataset $\{(x_i, y_i),\ i = 1, 2, \cdots, n\}$.
- Find

$$\hat{f}(X) = \arg\min_{f \in \mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq f(x_i))$$

- This training error rate quantifies the proportion of mistakes (misclassfications) that are made if we use $f(x_i)$ to predict labels of y_i to the training observations.
- $\hat{f}$ is the classifier that minimizes the misclassification rate over class $\mathcal{C}$.
- The test (validation) misclassification rate is

$$\text{Error Rate}_{Valid}(\mathcal{C}) = \frac{1}{m} \sum_{j=1}^{m} I(y_{0,j} \neq \hat{f}(x_{0,j}))$$

where $\{(x_{0,j}, y_{0,j}),\ j = 1, 2, \cdots, m\}$ is a test (validation) dataset.

Choice of $\mathcal{C}$: Logistic Regression

- The labels for Y are assumed to be 0 and 1 instead of a and b. Assume for now that there is one independent variable X.
- Logistic regression assumes that

$$\pi(1|x, \underline{\beta}) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

where $\underline{\beta} = (\beta_0, \beta_1)$ and

$$\pi(0|x, \underline{\beta}) = 1 - \pi(1|x, \underline{\beta}) = \frac{1}{1+e^{\beta_0+\beta_1 x}}$$

Based on the training criteria, we seek

$$\hat{f}(x\,;\,\hat{\underline{\beta}}) = \arg\min_{\underline{\beta}} \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq f(x_i\,;\,\underline{\beta}))$$

where

$$\begin{aligned} f(x\,;\,\underline{\beta}) &= 1 \quad \text{if } \pi(1|x,\underline{\beta}) > \pi(0|x,\underline{\beta}), \text{ and} \\ &= 0 \quad \text{if } \pi(1|x,\underline{\beta}) < \pi(0|x,\underline{\beta}). \end{aligned}$$

Difficulty in Minimization

- The minimization

$$\hat{f}(x;\hat{\underline{\beta}}) = \arg\min_{\underline{\beta}} \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq f(x_i;\underline{\beta}))$$

where

$$\begin{aligned} f(x;\underline{\beta}) &= 1 \quad \text{if } \pi(1|x,\underline{\beta}) > \pi(0|x,\underline{\beta}), \text{ and} \\ &= 0 \quad \text{if } \pi(1|x,\underline{\beta}) < \pi(0|x,\underline{\beta}). \end{aligned}$$

is impossible to do since $f(x;\underline{\beta})$ is not a continuous function of $\underline{\beta}$.

- $f(x;\underline{\beta})$ obtained from $\pi(1|x,\underline{\beta})$ above is called hard thresholding.
- We need a loss function that is continuously differentiable in $\underline{\beta}$. We use soft thresholding: We assume that $y_i \sim Ber(\pi(1|x_i,\underline{\beta}))$ independently for each $i = 1, 2, \cdots, n$.

The Logistic Loss (Log Loss) Function

- Since $y_i \sim Ber(\pi(1|x_i, \underline{\beta}))$ independently for each $i = 1, 2, \cdots, n$, the likelihood is given by

$$\ell(\beta_0, \beta_1 ; \underline{y}) = \prod_{i=1}^{n} [\pi(1|x_i, \underline{\beta})]^{y_i} [\pi(0|x_i, \underline{\beta})]^{1-y_i}$$

- The general relationship between a likelihood ℓ and its corresponding loss function L is

$$L = -\log(\ell)$$

- So, the loss function used to train the logistic regression model is

$$\begin{aligned} L_{Logistic}(\underline{\beta}) &= -\log \ell(\beta_0, \beta_1 ; \underline{y}) \\ &= \sum_{i=1}^{n} y_i \log \left[\pi(1|x_i, \underline{\beta})\right] + (1 - y_i) \log \left[\pi(0|x_i, \underline{\beta})\right]. \end{aligned}$$

- This is called the logistic loss function (log loss) and is used in place of the misclassification loss function to estimate $\underline{\beta}$.

Classification based on Log Loss: Steps Involved

- Choose the class

$$\mathcal{C} = \left\{ \pi(1|x, \underline{\beta}) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \right\}$$

- Minimize the log loss with respect to $\underline{\beta}$ over a training dataset:

$$(\hat{\beta}_0, \hat{\beta}_1) = \arg\min_{\underline{\beta}} \frac{1}{n} \sum_{i=1}^{n} y_i \log\left[\pi(1|x_i, \underline{\beta})\right] + (1-y_i) \log\left[\pi(0|x_i, \underline{\beta})\right].$$

- Generate the classifier

$$\begin{aligned} \hat{f}(x) \equiv f(x\,;\,\underline{\hat{\beta}}) &= 1 \quad \text{if } \pi(1|x, \underline{\hat{\beta}}) > \pi(0|x, \underline{\hat{\beta}}), \text{ and} \\ &= 0 \quad \text{if } \pi(1|x, \underline{\hat{\beta}}) < \pi(0|x, \underline{\hat{\beta}}) \end{aligned}$$

- Obtain the outcomes of the classifier on a test (validation) dataset and compute

$$\text{Error Rate}_{Valid}(\mathcal{C}) = \frac{1}{m} \sum_{j=1}^{m} I(y_{0,j} \neq \hat{f}(x_{0,j}))$$

Example

- Consider the `Default` data set where the response $Y =$ `default` falls into one of two categories, `Yes` or `No`.
- We will use logistic regression to model the probability that Y belongs to a particular category.

- For the `Default` data, logistic regression models the probability of `default` given `balance` as

$$\pi(\texttt{default} = \texttt{Yes}|\texttt{balance}) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

  where $X =$ `balance`.

- The values of $\pi(\texttt{default} = \texttt{Yes}|\texttt{balance})$ will range between 0 and 1.
- Then for any given value of `balance`, a prediction can be made for `default`.
- For example, one might predict `default` $=$ `Yes` for any individual for whom $\pi(\texttt{default} = \texttt{Yes}|\texttt{balance}) > 0.5$

Training, Testing and CV with R codes

- Logistic regression can be easily fit using R, and so there is no need to go into the details of the maximum likelihood fitting procedure.
- To fit logistic regression, we use the `glm()` function that fits a variety of generalized linear models in R including logistic regression.
- The syntax of `glm()` is similar to that of `lm()` but an additional argument `family = binomal` has to be given to run logistic regression instead of another type of generalized linear model.
- Note that the strcture of the R codes for training, testing and CV are all the same as previously.
- The only differences are to replace `lm()` by `glm()` in the training part, and
- To replace the squared error loss function by the $0-1$ loss function in the testing/validation part.

Example:

Do the following:

- Fit a logistic regression model that is linear in x in the exponent, that is,

$$\pi(\texttt{default} = \texttt{Yes} | \texttt{balance}) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}.$$

Call this class $\mathcal{C}_1$.

- Next, similar to polynomial regression, fit a logistic regression model that is a polynomial of degree p in x in the exponent, that is,

$$\pi(\texttt{default} = \texttt{Yes} | \texttt{balance}) = \frac{e^{\beta_0 + \sum_{j=1}^{p} \beta_j x^j}}{1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j x^j}}.$$

Call this class $\mathcal{C}_p$.

- Run the CV procedure for $1 \leq p \leq 7$ and choose the best p^* that minimizes the CV error rate.

Here are the R codes

```
#Unit 2 of ML
#install.packages("ISLR")
#This R package contains all datasets
#used in the book
library(ISLR)
#Attach the Default dataset to R workspace
attach(Default)
str(Default) #Gives to the structure of the DF
```

```
## 'data.frame': 10000 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1
##  $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

R codes (cont.)


```
logistic_fit <-
glm(default ~ balance, family = binomial,
data = Default)
summary(logistic_fit)
```

R codes (cont.)

```
##
## Call:
## glm(formula = default ~ balance, family = binomial, data =
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
##
## (Dispersion parameter for binomial family taken to be 1)
```

R codes (cont.)

```
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```



```
#Get predicted values for balance values
#in the Default dataset
#The syntax is the same as before but with an optional
#argument type, default type is on the scale of the linear
#predictors. type="response" gives probabilities in the
#scale of the response variable
pred_probabilities <- predict(logistic_fit,
 newdata = Default, type="response")
predicted_classes <- ifelse(pred_probabilities> 0.5,
 "Yes", "No")
```

R codes (cont.)

```
# Model Accuracy
with(Default,
mean(predicted_classes == default)
)
```

```
## [1] 0.9725
```

Note that if you want to get model misclassification error, either do one minus 0.9725 or

```
#If you want to get Model Misclassification Error Rates
with(Default,
mean(predicted_classes != default)
)
```

```
## [1] 0.0275
```

R codes for CV


```
#Now do CV
#Now let's do full CV
library(dplyr)

K = 50;
P = 3;
MCE_train_mat <- rep(list(vector("numeric",K)),P)
MCE_valid_mat <- rep(list(vector("numeric",K)),P)
glm_deviance <- rep(list(vector("numeric",K)),P)

for (k in 1:K){
#Training dataset data.frame
train <- Default %>% sample_frac(0.7)
#Validation dataset data.frame
valid <- Default %>% setdiff(train)
```

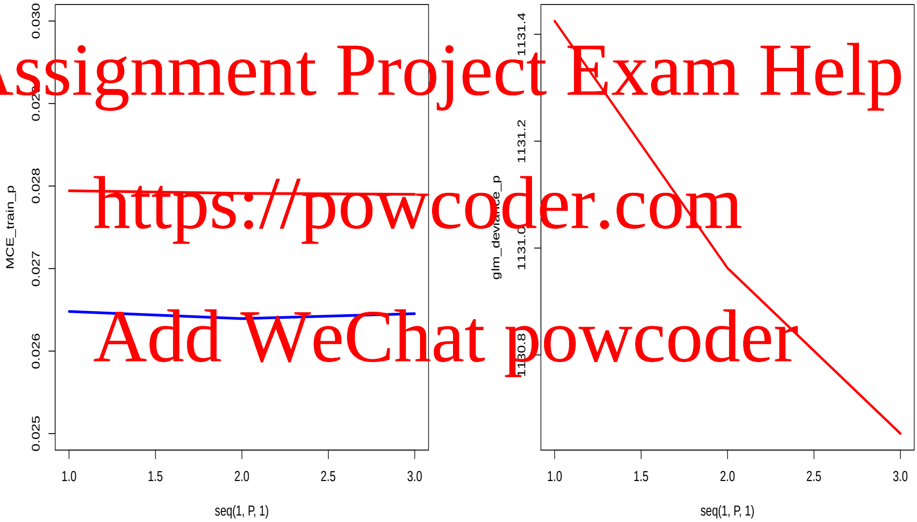
R codes for CV

```
#Determine class of learners which are polynomails
#from degree 1 to 3
for (p in 1:P){
poly_train_fit <- glm(default ~ poly(balance,p),
data = train, family = binomial)
glm_deviance[[p]][k] <- poly_train_fit$deviance

poly_train_predict_probs <- predict(
poly_train_fit, train, type="response")
predicted_classes_train <- ifelse(
poly_train_predict_probs > 0.5, "Yes", "No")

poly_valid_predict_probs <- predict(
poly_train_fit, valid, type="response")
predicted_classes_valid <- ifelse(
poly_valid_predict_probs > 0.5, "Yes", "No")
```

R codes for CV

```
MCE_train_mat[[p]][k] <- with(train,
 mean(default != predicted_classes_train))
MCE_valid_mat[[p]][k] <- with(valid,
 mean(default != predicted_classes_valid))
}
}
MCE_train_p <- sapply(MCE_train_mat, mean)
MCE_valid_p <- sapply(MCE_valid_mat, mean)
plot(seq(1,P,1), MCE_train_p, col="red",
 type="l", lwd=3, ylim = c(0.025, 0.03))
lines(seq(1,P,1), MCE_valid_p, type="l",
col="blue", lwd=3)

glm_deviance_p <- sapply(glm_deviance, mean)
plot(seq(1,P,1), glm_deviance_p, col="red", type="l", lwd=3)
```

Plot of Training and Validation Error Rates



Logistic Regression for Multiple Explanatory Variables

- Assume that there are p independent variables $X_1, X_2, \cdots, X_p$.
- Logistic regression assumes that

$$\pi(1|x, \underline{\beta}) = \frac{e^{\beta_0 + \sum_{j=1}^{p} \beta_j x_j}}{1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j x_j}}$$

where $\underline{\beta} = (\beta_0, \beta_1, \cdots, \beta_p)$ and

$$\pi(0|x, \underline{\beta}) = 1 - \pi(1|x, \underline{\beta}) = \frac{1}{1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j x_j}}$$

Based on the training criteria, we seek

$$\hat{f}(x\,;\,\hat{\underline{\beta}}) = \arg\min_{\underline{\beta}} \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq f(x_i\,;\,\underline{\beta}))$$

where

$$\begin{aligned} f(x\,;\,\underline{\beta}) &= 1 \quad \text{if } \pi(1|x, \underline{\beta}) > \pi(0|x, \underline{\beta}), \text{ and} \\ &= 0 \quad \text{if } \pi(1|x, \underline{\beta}) < \pi(0|x, \underline{\beta}). \end{aligned}$$