Statistical Computing with R: Masters in Data Sciences 503, S21 First Batch, SMS, TU, 2021

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Data Analysis and Decision Modeling, MBA, Pokhara University, Nepal

Faculty, FAIMER Fellowship in Health Professions Education, India/USA.

Review Preview

- Simple Linear Regression
  - Gradient Decent fit
  - Model Accuracy
  - Model Prediction
  - Model Validation
    - Validation set, Leave one out cross validation, k-folds cross validation, repeated k-folds cross-validation etc.)

- Multiple Linear Regression
  - Simple linear regression +
  - Multicollinearity, its assessment and solutions
  - Regularization
    - Ridge
    - Lasso
    - Elastic Net (Ridge+Lasso)

Simple Linear Regression: Gradient Decent

https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

- Linear Regression can also be fitted with Gradient Decent algorithm instead of OLS

- Here we minimize the loss function (E) to find m (slope b) and c (constant a)

$$E = \frac{1}{n} \sum_{i=0}^{n} (y_i - (mx_i + c))^2$$

E = MSE = Mean Sum of Square

$$D_m = \frac{1}{n} \sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

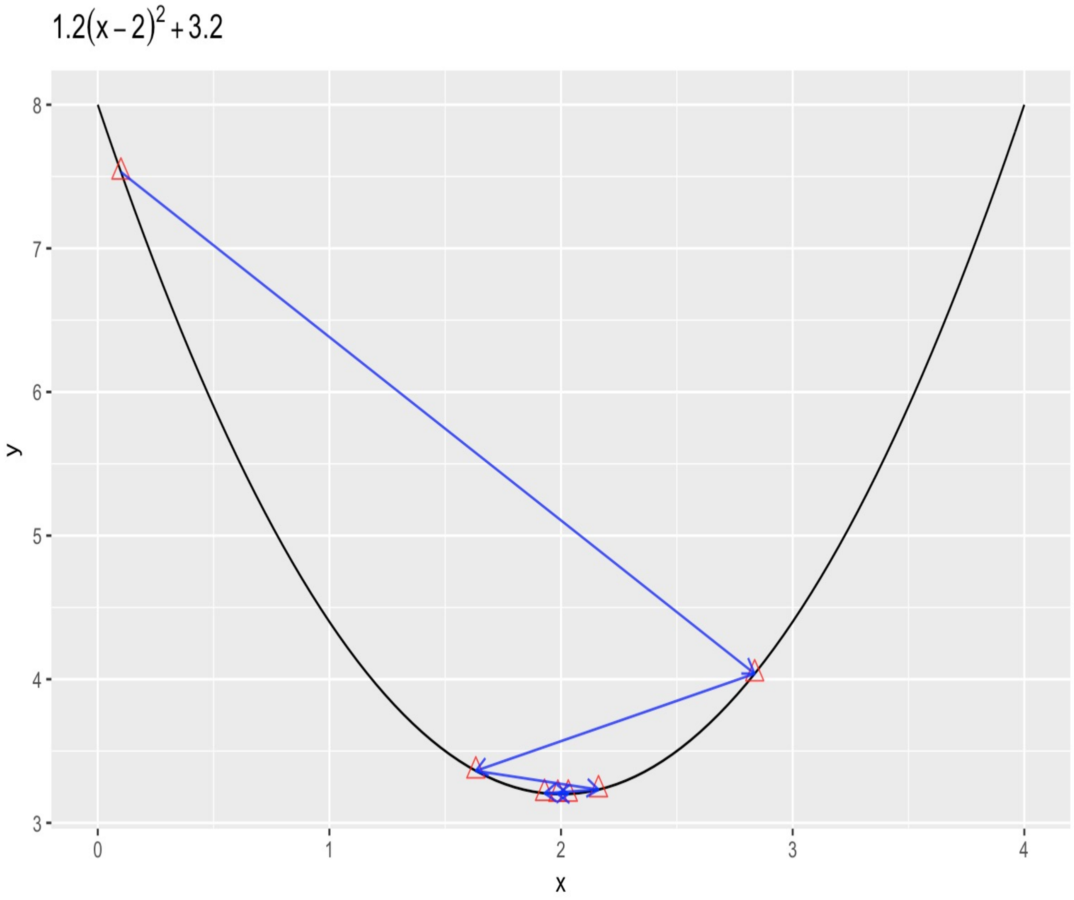
$$D_m = \frac{-2}{n} \sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

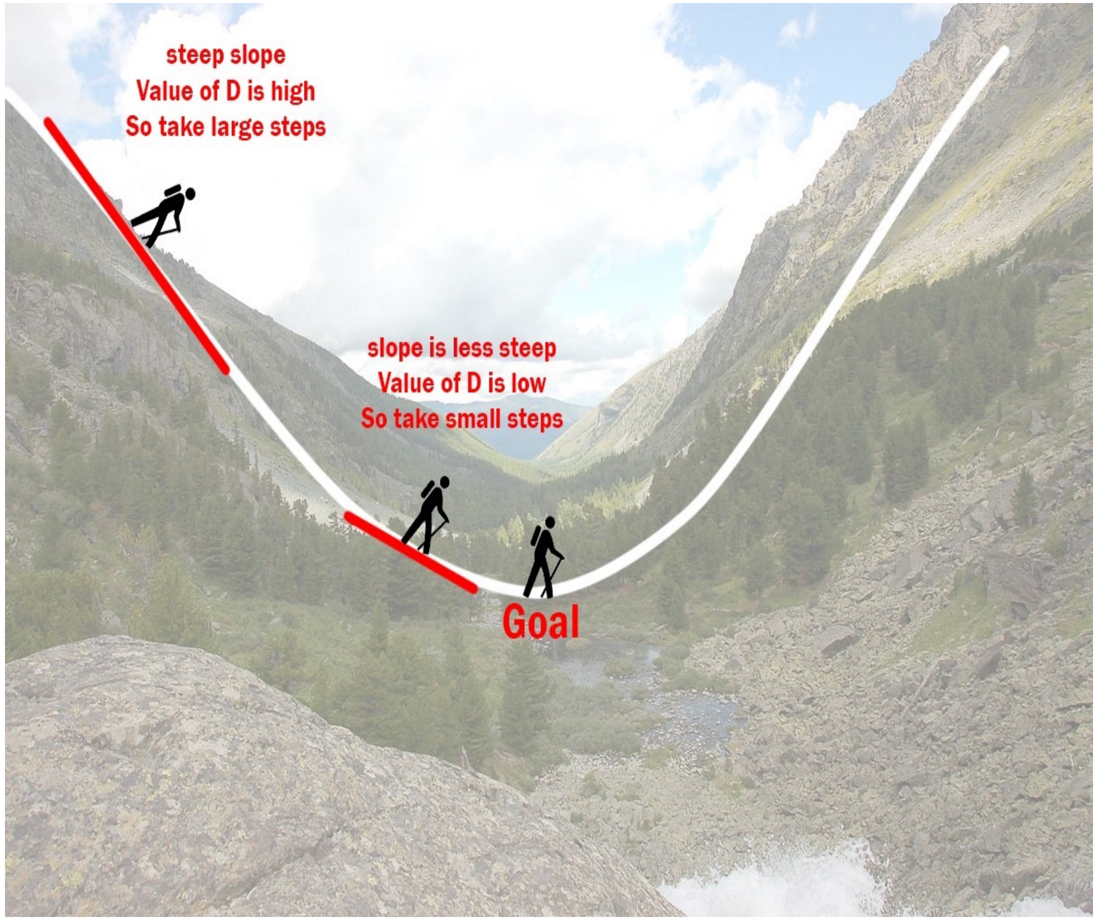
$$D_c = \frac{-2}{n} \sum_{i=0}^{n} (y_i - \bar{y}_i)$$

We can solve these two equations to get the value of m and c. This is an optimization problem as we need to MINIMIZE the loss function MSE using multiple iterations!

Illustration:

http://ethen8181.github.io/machine-learning/linear_regression/linear_regession.html





How to get MSE of Simple Linear Model? (OLS and Gradient Decent)

- Using the residuals of linear model:

```
lm1 <- lm(mpg ~ wt, data=mtcars)
(mse <- mean(lm1$residuals^2))
```

- Saving predicted values:

```
data <- data.frame(pred = predict(lm1), actual = mtcars$mpg)
head(data)
mean((data$actual - data$pred)^2)
```

- > (mse <- mean(lm1$residuals^2))
- **[1] 8.697561**

- > mean((data$actual - data$pred)^2)
- **[1] 8.697561**

Model Accuracy of Linear Model:

- **R-square** – Explained variance (higher is better!)

- **RMSE** – Root of MSE (lower is better)
- MAE – Mean Absolute Error (lower is better)

- MAPE – Mean Absolute Percentage Error (lower is better)

$$MAPE = \frac{\overset{\text{Multiplying by 100\% converts to percentage}}{\boxed{100\%}}}{n} \sum \left| \frac{\overbrace{y - \hat{y}}^{\text{The residual}}}{\underset{\text{Each residual is scaled against the actual value}}{\boxed{y}}} \right|$$

#Better to use "caret" package

Install.packages("caret")

library(caret)

R2 <- R2(data$pred, data$actual)

- 0.7528328

RMSE <- RMSE(data$pred, data$actual)

- 2.949163

MAE <- MAE(data$pred, data$actual)

- 2.340642

Get MAPE in R as:

- 12.60733

If LINE is valid after BLUE then we can predict:

(So, we will use "lm1<-lm(mpg~wt, data=mtcars)" model to do it!)

- We need to save independent variable value/values in a new data:

```
new.wt <- data.frame(wt = 6)
```

- We can then use this data to predict the value of the dependent variable based on the fitted model as:

```
predict(lm1, newdata = new.wt)
```

Result = 5.218297

- Interpretation: Cars with 6000 lbs weight will (only) give 5.22 miles per gallon as per the linear regression algorithm!)

Validation & Cross-validation for Predictive Modelling including Linear Model:

- In statistics, we normally use the "full" data to do predictions

- In machine learning, we use validation/cross-validation sets to do the predictions

- Validation/Cross-validation can be done with:
  - Validation set (data split)
  - Leave one out cross-validation (LOOCV)
  - K-fold cross-validation
  - Repeated k-fold validation

Validation: Validation set (widely used!)

- Here the full data is “randomly” divided into two sets:
  - Training set
  - Testing set (validation set)

- Then model is fitted in the training set

- The model fit is then validated in the testing set using prediction

- This is the most widely used cross-validation method in the supervised machine learning

Lets do it for “mtcars” data:

#Define the mtcars data as “data”:

data <- mtcars

#Use random seed to replicate the result

set.seed(1234)

#Do random sampling to divide the cases into two independent samples

ind <- sample(2, nrow(mtcars), replace = T, prob = c(0.7, 0.3))

#Data partition

train.data <- data[ind==1,]

test.data <- data[ind==2,]

Model Fit, Prediction and Cross-Validation: Validation set approach

```
lm4 <- lm(mpg~wt, data = train.data)
library(dplyr)
library(caret)
predictions <- lm4 %>% predict(test.data)
data.frame(R2 = R2(predictions, test.data$mpg),
           RMSE = RMSE(predictions, test.data$mpg),
           MAE = MAE(predictions, test.data$mpg))
```

Model Accuracy of Training dataset:

summary(lm4)

- **Multiple R-squared: 0.7013**
- MSE = 9.526359 (How?)
- RMSE = SQRT(MSE) = 3.08648

Model Accuracy of Testing dataset:

R2	RMSE	MAE
0.9031085	**2.279303**	**1.698583**

Model Fit, Prediction and Cross-Validation: Leave-One-Out Cross-Validation approach:

```
#Leave one out CV
library(caret)

# Define training control
train.control <- trainControl(method = "LOOCV")

# Train the model
model1 <- train(mpg ~wt, data = mtcars, method = "lm",
            trControl = train.control)

# Summarize the results
print(model1)
```

```
Linear Regression
32 samples
 1 predictor

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 31, 31, 31, 31, 31, 31, ...
Resampling results:

 RMSE       Rsquared    MAE
 3.201673   0.7104641   2.517436

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Prediction with LOOCV:

```
predictions1 <- model1 %>% predict(test.data)
data.frame(R2 = R2(predictions1, test.data$mpg),
           RMSE = RMSE(predictions1, test.data$mpg),
           MAE = MAE(predictions1, test.data$mpg))
```

R2	RMSE	MAE
0.9031085	2.244232	1.714515

Model Fit, Prediction and Cross-Validation: K-folds Cross-Validation approach

```
#k-fold cross validation
library(caret)
# Define training control
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)

# Train the model
model2 <- train(mpg ~ wt, data = mtcars, method = "lm",
            trControl = train.control)

# Summarize the results
print(model2)
```

```
Linear Regression
32 samples
 1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 28, 28, 29, 29, 29, 30, ...
Resampling results:

 RMSE       Rsquared    MAE
 2.85133    0.7346939   2.375068

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Predictions with k-folds CV:

```
predictions2 <- model2 %>% predict(test.data)
data.frame(R2 = R2(predictions2, test.data$mpg),
           RMSE = RMSE(predictions2, test.data$mpg),
           MAE = MAE(predictions2, test.data$mpg))
```

R2	RMSE	MAE
0.9031085	2.244232	1.714515

Model Fit, Prediction and Cross-Validation: Repeated K-folds Cross-Validation approach

```
#repeated k-fold cross validation
library(caret)
# Define training control
set.seed(123)
train.control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)
# Train the model
model <- train(mpg ~wt, data = mtcars, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

```
Linear Regression
32 samples
 1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 28, 28, 29, 29, 29, 30, ...
Resampling results:

  RMSE       Rsquared    MAE
  2.975392   0.8351572   2.539797
Tuning parameter 'intercept' was held constant at a value of TRUE
```

Prediction with repeated k-folds CV:

predictions3 <- model3 %>% predict(test.data)

data.frame(R2 = R2(predictions3, test.data$mpg),

RMSE = RMSE(predictions3, test.data$mpg),

MAE = MAE(predictions3, test.data$mpg))

R2	RMSE	MAE
0.9031085	2.244232	1.714515

Summary: Which one should be used based on R-squared values of “lm” model?

- R-square for training set: 0.7013
- R-square for training with LOOCV: 0.7104641
- R-square for training with k-folds CV: 0.7346939
- R-square for training with repeated k-folds CV: 0.8351572

- R-square for testing set: 0.9031085
- R-square for testing with LOOCV: 0.9031085
- R-square for testing with k-folds CV: 0.9031085
- R-square for testing with repeated k-folds CV: 0.9031085

Summary: Which one should be used based on RMSE value?

- RMSE for training set: 3.08648
- RMSE for training with LOOCV : 3.201673
- RMSE for training with k-folds CV: 2.85133
- RMSE for training with repeated k-folds CV: 2.975392

- RMSE for testing test: 2.279303
- RMSE for testing with LOOCV: 2.244232
- RMSE for testing with k-folds CV: 2.244232
- RMSE for testing with repeated k-folds CV: 2.244232

Quick Think!

- Which model must be selected: Based on R-square or based on RMSE?
- Do we need to check the BLUE and LINE assumptions for the fit done with the training data?
- If BLUE and LINE test is a must then which training model should be checked?
- Validation set model?
- LOOCV set model?
- K-fold CV set model?
- Repeated K-fold CV set model?

Question/queries so far?

Multiple linear regression:

- It is an extension of the simple linear regression
- Multiple linear regression have more than one (two or more) independent variables
- Multiple linear regression has one (1) continuous dependent variable so it is a supervised learning
- All the assumptions of the simple linear regression are also applicable here
- There is one more condition:
- Multicollinearity must not be present i.e. correlations between independent variables must not be "high"

Assessing multicollinearity:

- Pearson correlation coefficients can be used
- We need to get a correlation matrix and flag the correlations with more than 0.75
- These pair/s of independent variables influences the linear model coefficients

- Variance Inflation Factor (VIF) is most commonly used to assess multicollinearity
- We can get the VIF for each independent variable
- Multicollinearity will be confirmed for an independent variable with VIF > 10 for linear models

Fitting multiple linear regression model using "mtcars" data:

- mlr <- lm(mpg ~., data = mtcars)
- summary(mlr)
- library(car)
- vif(mlr)
- We need to drop the independent variable with highest VIF and run the model again until all the VIF <10!

- None of the variables used in the model are statistically significant!

> vif(mlr)

cyl	**disp**	hp
15.373833	**21.620241**	9.832037
drat	wt	qsec
3.374620	15.164887	7.527958
vs	am	gear
4.965873	4.648487	5.357452
carb		
7.908747		

Fitting multiple linear regression using “mtcars” data:

#Removing “disp” variable:

```
mlr1 <- lm(mpg ~ cyl+hp+drat+wt+qsec+vs+am+gear+carb, data = mtcars)
summary(mlr1)
vif(mlr1)
```

- We need to drop the independent variable with highest VIF and run the model again until all the VIF <10!

- The “wt” variable is significant

- > vif(mlr1)

cyl	hp	drat
14.284737	7.123361	3.329298
wt	qsec	vs
6.189050	6.914423	4.916053
am	gear	carb
4.645108	5.324402	4.310597

Fitting multiple linear regression using "mtcars" data:

#Removing "cyl" variable:

mlr2 <- lm(mpg ~ hp+drat+wt+qsec+vs+am+gear+carb, data = mtcars)

summary(mlr1)

vif(mlr1)

- We need to drop the independent variable with highest VIF and run the model again until VIF <10!
- **If all the VIF < 10 then we can interpret the model and do the predictions**

- The "wt" variable is significant, b = -2.60968 (1 unit increase in wt reduces the mpg by 2.61 unit controlling for other independent variables)

- > vif(mlr2)

hp	drat	wt
6.015788	3.111501	6.051127
qsec	vs	am
5.918682	4.270956	4.285815
gear	carb	
4.690187	4.290468	

Assignment:

- Use the validation and cross-validation methods for the multiple linear regression (mlr2) model
- Which model is the best model?
- Why?
- Predict the weight of the cars based on the best model identified using the test.data
- Change all the variables (except mpg) as standardized variable using “scale” command in R/R Studio
- Fit the multiple linear regression model with these standardized variables
- **Does it solve the multicollinearity issue?**
- Why? Write conclusions.

Alternative way to deal with multicollinearity in data science/machine learning:

- We can use the “regularization” methods

- The most common ones are:

    - Ridge regression
    - Lasso regression
    - Elastic net regression

- Once the “multicollinearity” problem is fixed then we can do the predictions and use the validation indices to select the best model for our data!

- **I will post a YouTube video link and you can fit these models for the “mtcars” data and learn from them!**

Question/queries?

Next class

- Other regression models used in the supervised learning: polynomial regression, KNN algorithm etc.

- Logistic regression

- Other classification models used in the supervised learning: Naïve Bayes, Decision Trees, SVM etc.

Thank you!

@shitalbhandary