Week9
Simple Linear Regression
Gradient Descent

Zhe Zhao

Stevens Institute of Technology

zzhao6@stevens.edu

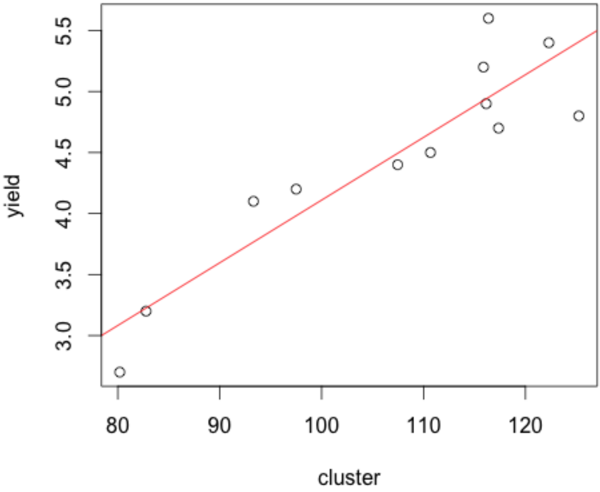
November 2, 2015

Outline

1. Simple Linear Regression
2. Gradient Descent
3. Project Ideas

Simple Linear Regression

Simple Linear Regression



Simple Linear Regression

Objective

Describe the relationship between two variables, say X and Y as a straight line, that is, Y is modeled as a linear function of X.

Variables

X: explanatory variable

Y: response variable

After data collection, we have pairs of observations:

$$(x_1, y_1), ..., (x_n, y_n)$$

Simple Linear Regression

Model

Then we fit our data set into this simple linear model.

$$Y = \alpha + \beta X + \epsilon, \text{ where } i = 1, 2, ..., n$$

- Residuals: $\epsilon_i \sim N(0, \sigma^2)$, independent
- Estimate y_j by the value of x_j, $\hat{y}_j = \alpha + \beta x_j = E(y_j | x_j)$

Parameters

We need to figure out a way to estimate these parameters:

- α (Intercept): point in which the line intercepts the y-axis
- β (Slope): increase in Y per unit change in X

Simple Linear Regression

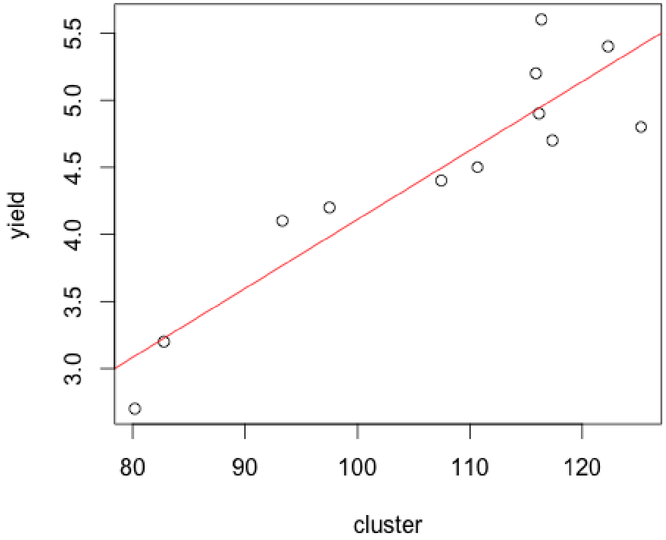
Example

```
> yield <- data$yield
> cluster <- data$cluster.count
> plot(yield ~ cluster)
> lm(yield ~ cluster)

Call:
lm(formula = yield ~ cluster)

Coefficients:
(Intercept)      cluster
   -1.02790      0.05138

> lm.r <- lm(yield ~ cluster)
> abline(lm.r, col='red')
```


yield
5.5
5.0
4.5
4.0
3.5
3.0
80
90
100
110
120
cluster

Example

```
> summary(lm.r)
Call:
lm(formula = yield ~ cluster)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60700 -0.19471 -0.03241  0.23220  0.64874

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.02790    0.78355  -1.312    0.219
cluster      0.05138    0.00725   7.087 3.35e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
Residual standard error: 0.3641 on 10 degrees of freedom
Multiple R-squared:  0.834, Adjusted R-squared:  0.8174
F-statistic: 50.23 on 1 and 10 DF,  p-value: 3.347e-05
```

Different Types of Linear Models in R

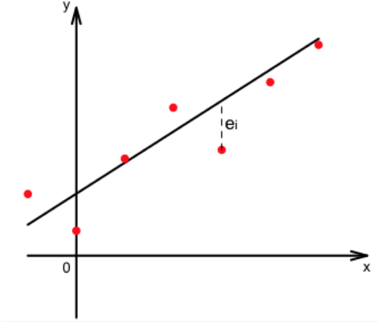
Syntax	Model
$Y \sim X$	$Y = \alpha + \beta X$
$Y \sim -1 + X$	$Y = \beta X$
$Y \sim X + I(X^2)$	$Y = \alpha + \beta_1 X + \beta_2 X^2$
$Y \sim X_1 + X_2$	$Y = \alpha + \beta_1 X_1 + \beta_2 X_2$
$Y \sim X_1 : X_2$	$Y = \alpha + \beta X_1 X_2$
$Y \sim X_1 * X_2$	$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$

Least Squares

We want to find the equation of the line that best fits the data. It means finding α and β such that the fitted values of y_j , given by

$$\hat{y}_j = \alpha + \beta x_j$$

are as close as possible to the observed values y_i, for all $i = 1, 2, ..., n$.



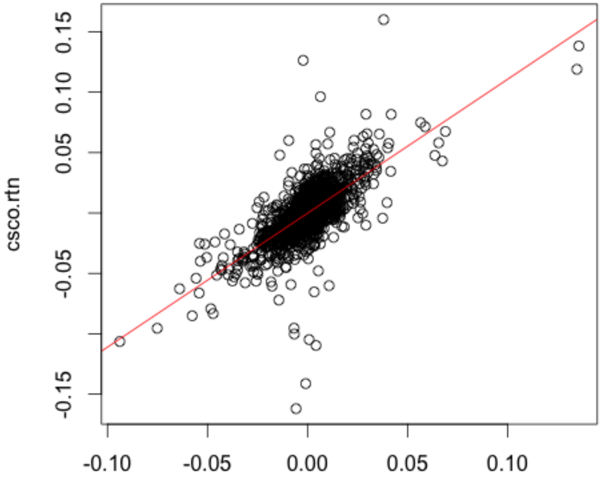
residuals given by:
$\epsilon_i = y_i - \hat{y}_i$

Example on Stock Returns

Example

```
> getSymbols("CSCO")
> getSymbols("DIA")
> csco <- data.frame(CSCO)
> dia <- data.frame(DIA)
> # get last price
> csco.price <- csco$CSCO.Adjusted
> dia.price <- dia$DIA.Adjusted
> # get returns
> csco.rtn<-diff(csco.price, lag = 1)/
          csco.price[-length(csco.price)]
> dia.rtn<-diff(dia.price, lag = 1)/
          dia.price[-length(dia.price)]
> plot(csco.rtn ~ dia.rtn)
> lm1 <- lm(csco.rtn ~ dia.rtn)
> abline(lm1, col = 'red')
```

Example on Stock Returns



Least Squares

Estimation of Parameters

A usual way of calculating α and β is based on the minimization of the sum of the squared residuals, or residual sum of squares (RSS):

$$RSS = \Sigma_{i=1}^{n} \epsilon_i^2$$

$$= \Sigma_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \Sigma_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

Least Squares

- In fact, we can derive the formula for α and β given certain data set. It is called normal equation:
$$\theta = (X^T X)^{-1} X^T Y$$
where $\theta = (\alpha, \beta)$

- Another way is to use numerical algorithms calculating the values of α and β: Gradient Descent, Newton's Method.

Gradient Descent

Gradient

Definition

If we have a bivariate function $f(x, y)$, then the partial derivatives, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are the rate of change of f with respect to x and y.

We put them together in a vector, and call it *Gradient of f*:

$$\nabla f = \langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \rangle$$

Gradient

- Of course, if we specify a point P_0 and we can calculate the gradient on that point:

$$\nabla f|_{P_0} = \langle \frac{\partial f}{\partial x}|_{P_0}, \frac{\partial f}{\partial y}|_{P_0} \rangle$$

- For functions with only one variable, the gradient equals to the derivative.
- Similarly, for higher dimensional functions, for example $f(x_1, ..., x_n)$, we have:

$$\nabla f = \langle \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \rangle$$

Gradient Descent Algorithm

Algorithm: Single Variate Function

```
loop {
```

$$x := x - \alpha \frac{df}{dx}$$

```
    # until converge
}
```

Algorithm: Multi Variate Function

```
loop {
```

$$\mathbf{x} := \mathbf{x} - \alpha \nabla f$$

```
    # until converge
}
```

where α is named “step size” or “learning rate”.

Gradient Descent Algorithm

Analysis:

- if $df/dx > 0$, which means $x_0 > x_{min}$, $x - \alpha\frac{df}{dx} \downarrow$
- if $df/dx < 0$, which means $x_0 < x_{min}$, $x - \alpha\frac{df}{dx} \uparrow$
- x_0 will converge to the extrema in no matter which case.
- The step size α also determines the speed of convergence.

Gradient Descent in R

Let's see how to implement this algorithm in R.

Example (pseudo code)

```
# set initial
# set convergence condition
# loop:
while(1)
{
    # algorithm
    # ...
    # if (converge)
    # {
    #    break
    # }
}
```

Gradient Descent in R

First define the function to be optimized: $f(x) = x^2 - 10x + 3$

Example

```
> # function f(x) = x^2 - 10x + 3
> # x.min = -b/2a = 5
> fx <- function(x) {
+     y = x^2 - 10 * x + 3
+     return (y)
+ }
> # derivative of f(x)
> # df = x^2 - 10
> df <- function(x) {
+     y = x*2 - 10
+     return (y)
+ }
```

Gradient Descent in R

Example

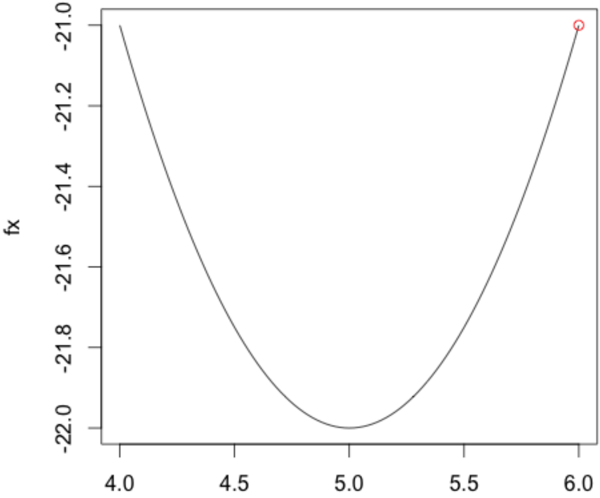
```
> plot(fx, xlim = c(4, 6))

> x0 = 6        # initial value
> points(x0, fx(x0), col = "red")

> alpha = 0.2   # step length

> epsilon = 0.0001 # condition to terminate the algorithm
> # step count
> step = 1
```

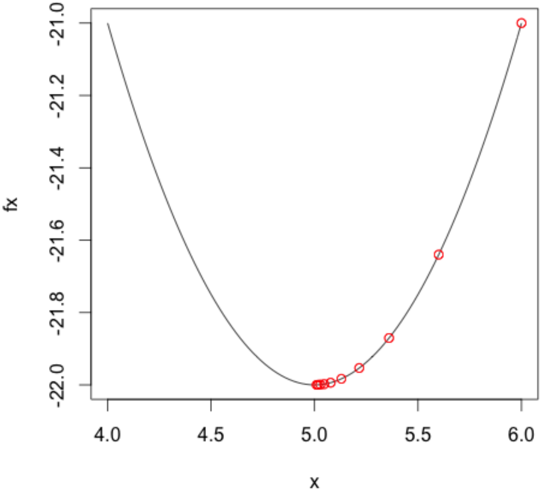
Gradient Descent in R



Example

```
> while(1)
+ {
+     cat("Calculating, step ", step, '\n', sep = "")
+     x1 = x0 - alpha * df(x0)    # update x
+
+     # check convergence
+     if (abs(fx(x1) - fx(x0)) < epsilon)
+     {
+         cat("x = ", x1, '\n', sep='')
+         cat("Final step: ", step, '\n', sep='')
+         break
+     }
+     points(x1, fx(x1), col = "red")
+     x0 = x1
+     step = step + 1
+     Sys.sleep(1.2) # suspend for a while
+ }
```

Gradient Descent in R



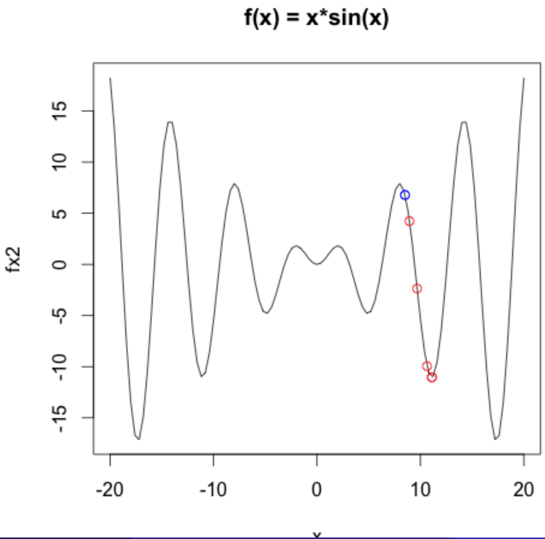
Gradient Descent in R

Two potential issue with this algorithm

- The size of α
- Local extrema.

Local Extrema

Local extrema, see details in the code



Gradient Descent and Linear Regression

Recall that we use *least squares* method to determine the parameters in linear regression.

$$RSS = \Sigma_{i=1}^{n} \epsilon_i^2$$

$$= \Sigma_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \Sigma_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

Thus our target is to minimize:

$$J(\beta) = \frac{1}{2} \Sigma_i (y_i - \beta^T X)^2, \text{ where } \beta = (\beta_0, \beta_1)^T$$

Gradient Descent and Linear Regression

Keep in mind that our algorithm is: $\beta := \beta - \alpha \frac{dJ}{d\beta}$, thus we need to calculate the gradient (derivative) w.r.t β.

Example

```
> x0 <- c(1,1,1,1,1) # column of 1's
> x1 <- c(1,2,3,4,5) # original x-values

> x <- cbind(x0,x1)
> y <- as.matrix(c(3,7,5,11,14))

> # number of elements
> m <- nrow(y)
```

Example

```
# define the gradient function dJ/d_beta:
# 1/m * (y^hat-y))*x where y^hat = x*beta
# in matrix form this is as follows:
> grad <- function(x, y, beta) {
+     gradient <- (1/m) * (t(x) %*% ((x %*% t(beta)) - y))
+     return(t(gradient))
+ }
# define gradient descent update algorithm
> grad.descent <- function(x, maxit){
+     beta <- matrix(c(0, 0), nrow=1) # Initialize
+
+     alpha = .05 # set learning rate
+     for (i in 1:maxit) {
+         beta <- beta - alpha  * grad(x, y, beta)
+     }
+     return(beta)
+ }
```

Project Ideas

R Packages

Packages Covered:

- lubridate
- fBasics
- quantmod
- TRTH
- Rmarkdown

Suggested Packages

- plyr: Tools for splitting, applying and combining data
- car: Applied Regression.
- ggplot2: An implementation of the grammar of graphics in R.
- ...

Connect to Bloomberg

- Package: Rbbg
    - Have to use on a Bloomberg Terminal
    - Dependencies: Java, rJava, Bloomberg Java API V3

Example

```
# sample 1
> conn <- blpConnect()
> bdp(conn, "AMZN US Equity", "NAME")

# sample 2
> securities <- c("AMZN US Equity", "OCN US Equity")
> fields <- c("NAME", "PX_LAST", "TIME", "SETTLE_DT",
"HAS_CONVERTIBLES")
> bdp(conn, securities, fields)
```

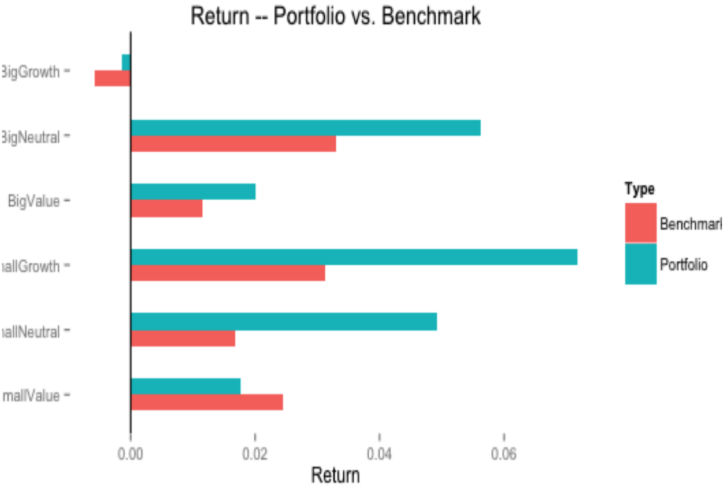
Connect to Database

- Package: RPostgreSQL
  - A database interface and PostgreSQL driver for R
  - It could let you connect to database and execute database command through R
  - Structured Query Language, PostgreSQL, MySQL, SQL Server

Portfolio Perfomance Analysis

- Package: pa
- Objective:
  - Create a portforlio based on the Fama & Frech style factors.
  - Apply Brinson model which is build in the package "pa" to analyze the Brinson.

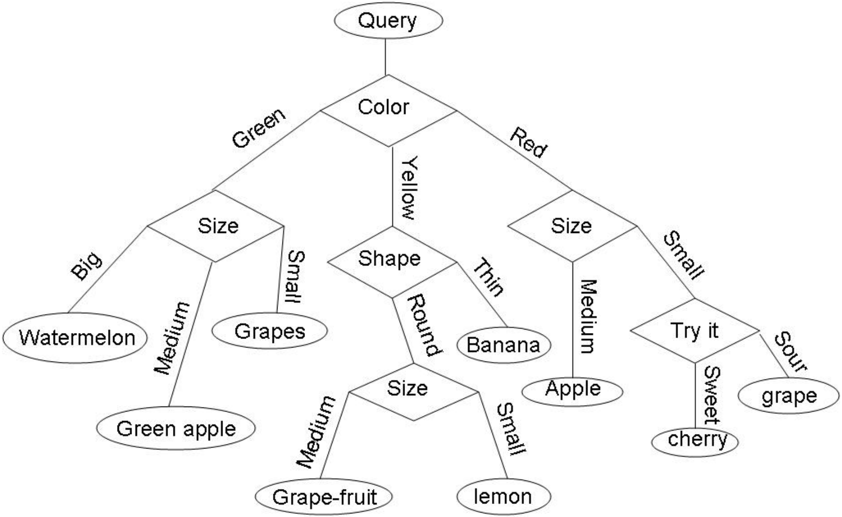
Portfolio Perfomance Analysis



Decision Trees

- Package: Party
  - Decision Trees are used as a classification tool in machine learning.
  - Data sets consist of large number of data points, where each data point is defined based on the set of the attributes and a label.
  - Decision Trees are a predictive method based on a branching series of Boolean tests

Decision Trees



Trading Indicators

- Package: TTR
  - TTR package serves as an excellent platform for developing and back-testing technical indicators.
  - Functions and data to construct technical trading rules with R
  - R topics documented: adjRatios, ADX, aroon, ATR, Bbands, CCI, chaikinAD, chaikinVolatility, CLV, CMF, CMO, DonchianChannel, DPO, DVI, EMV, GMMA, KST, lags, MACD, MFI, OBV, PBands, ROC, rollSFM, RSI, runPrecentRank, runSum, SAR, SMA, stoch, stockSymbols, TDI, TRIX, TTR, ttrc, VHF, volatility, williamsAD, WPR, ZigZag