Linear Regression

Art Tay

Gradient Derivation

a)

$$
\begin{aligned}
\frac{\partial}{\partial \beta_0}\left[\frac{1}{2m}\sum_{i=0}^{m}(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] &= \frac{\partial}{\partial \beta_0}\left[\frac{1}{2m}\sum_{i=0}^{m}(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] \\
&= \frac{1}{2m}\sum_{i=0}^{m}\left[2(\beta_0+\beta_1 x^{(i)}-y^{(i)})\cdot 1\right] \\
&= \frac{1}{m}\sum_{i=0}^{m}\beta_0+\beta_1\frac{1}{m}\sum_{i=0}^{m}x^{(i)}-\frac{1}{m}\sum_{i=0}^{m}y^{(i)} \\
&= \beta_0+\beta_1\bar{x}-\bar{y}
\end{aligned}
$$

b)

$$
\begin{aligned}
\frac{\partial}{\partial \beta_1}\left[\frac{1}{2m}\sum_{i=0}^{m}(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] &= \frac{1}{2m}\sum_{i=0}^{m}\frac{\partial}{\partial \beta_1}\left[(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] \\
&= \frac{1}{2m}\sum_{i=0}^{m}\left[2x^{(i)}(\beta_0+\beta_1 x^{(i)}-y^{(i)})\right] \\
&= \frac{1}{m}\sum_{i=0}^{m}\left[\beta_0 x^{(i)}+\beta_1 {x^{(i)}}^2-x^{(i)}y^{(i)}\right] \\
&= \beta_0\frac{1}{m}\sum_{i=0}^{m}x^{(i)}+\beta_1\frac{1}{m}\sum_{i=0}^{m}{x^{(i)}}^2-\frac{1}{m}\sum_{i=0}^{m}x^{(i)}y^{(i)} \\
&= \beta_0\bar{x}+\beta_1\bar{x^2}-\bar{xy}
\end{aligned}
$$

Linear Regression by Gradient Decent

```r
# Generates linear data with normal residuals
set.seed(123)
x <- rnorm(n = 30)

epsilon <- rnorm(n = 30)

y <- 5*x + 1 + epsilon
```

```r
summary(lm(y ~ x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6085 -0.5056 -0.2152  0.6932  2.0118
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1720     0.1534   7.639 2.54e-08 ***
## x             4.8660     0.1589  30.629  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8393 on 28 degrees of freedom
## Multiple R-squared:  0.971,  Adjusted R-squared:   0.97
## F-statistic: 938.1 on 1 and 28 DF,  p-value: < 2.2e-16
```

```r
# Calculates the mean squared error for a simple linear regression model.
# @param x - a vector of the explainatory variable.
# @param y - a vector of the response variable.
# @param beta_0 - the intercept value for the current SLR model.
# @param beta_1 - the slope value for the current SLR model.
# @return - sum total mean squared error (y_hat - y)^2
slr_mse <- function(x, y, beta_0, beta_1){
    cost <- ((beta_1 * x + beta_0) - y)^2
    return(sum(cost))
}
```

```r
# Calculates the slope and intercept values for SLR
# or simple linear regression.
# @param x - a vector of the explanatory variable.
# @param y - a vector of the response variable.
# @param alpha - the learning rate.
# @return betas - a vector containing the calculated betas.
slr_gradient_desc <- function(x, y, alpha){

    # Summary statistic calculations.
    # Helps to calculate the gradient faster.
    x_bar <- mean(x)
    y_bar <- mean(y)
    xy_bar <- mean(x*y)
    x_sqbar <- mean(x^2)

    # initial guess for beta_0 and beta_1.
    beta_0 <- y_bar
    beta_1 <- 0

    # A counter to determine is the error is unchanging.
    # This is the Loop-Control-Variable (LCV).
    count_same <- 0

    # Iterate 100 times or until the cost remains unchanged for 10 iterations.
```

```r
    for(i in 1:1000){

        # Stop the loop if the LCV >= 10.
        if(count_same >= 10){
            break
        }

        # Cost prior to beta adjustment.
        cost_start <- slr_mse(x, y, beta_0, beta_1)

        # Calculate gradient values.
        g_0 <- beta_0 + (beta_1 * x_bar) - y_bar
        g_1 <- (beta_0 * x_bar) + (beta_1 * x_sqbar) - xy_bar

        # Update betas.
        beta_0 <- beta_0 - (alpha * g_0)
        beta_1 <- beta_1 - (alpha * g_1)

        # If the cost is unchanged add 1 to the LCV.
        if(cost_start == slr_mse(x, y, beta_0, beta_1)){
            count_same <- count_same + 1
        }
    }

    return(c(beta_0 = round(beta_0, 4),
             beta_1 = round(beta_1, 4),
             iterations = i))
}
```

```r
slr_gradient_desc(x, y, alpha = 0.1)
```

```
##     beta_0     beta_1 iterations
##      1.172      4.866    256.000
```

Linear Model on Airbnb Data