# Linear Regression

Art Tay

## Gradient Derivation

a)

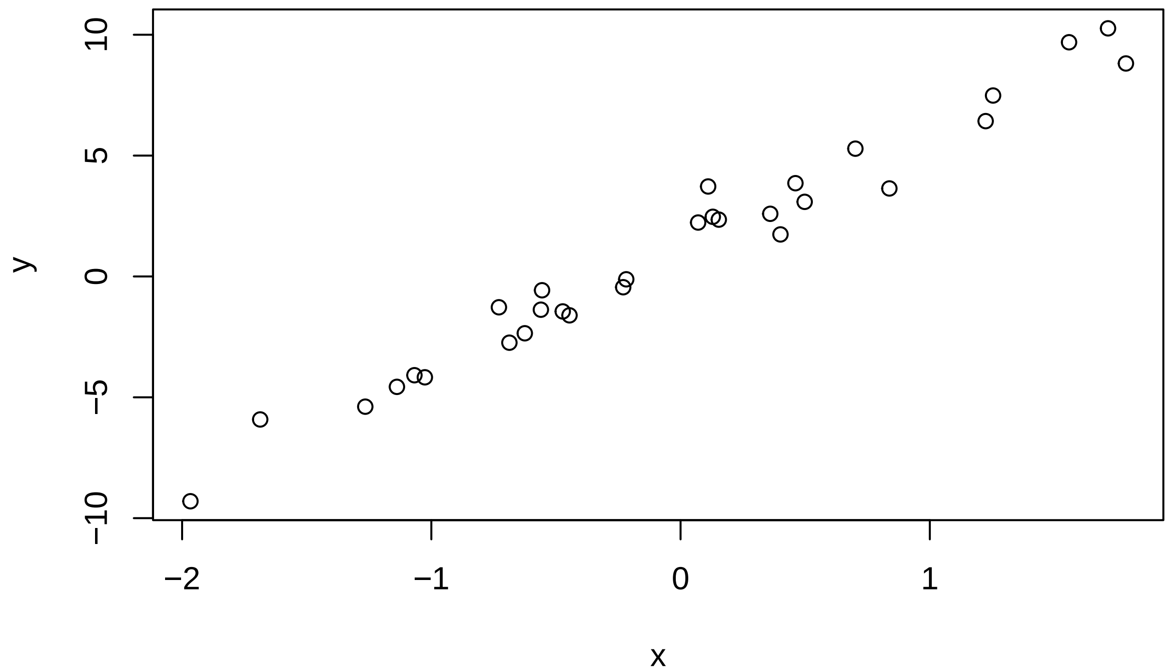
$$\begin{aligned}
\frac{\partial}{\partial \beta_0}\left[\frac{1}{2m}\sum_{i=0}^{m}(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] &= \frac{\partial}{\partial \beta_0}\left[\frac{1}{2m}\sum_{i=0}^{m}(\beta_0+\beta_1 x^{(i)}-y^{(i)})^2\right] \\
&= \frac{1}{2m}\sum_{i=0}^{m}\left[2(\beta_0+\beta_1 x^{(i)}-y^{(i)})\cdot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

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

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

epsilon <- rnorm(n = 30)

y <- 5*x + 1 + epsilon

plot(x,y)
```




```
ols_cost <- function(x, y, beta_0, beta_1){
    cost <- (y - (beta_1 * x + beta_0))^2
    return(sum(cost))
}
```

```
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

    cost_0 <- ols_cost(x, y, beta_0, beta_1)

    # Dataframe to store results.
    df_results <- c(beta_0 = beta_0,
                    beta_1 = beta_1,
                    error = cost_0)

    # A counter to determine is the error is unchanging.
```

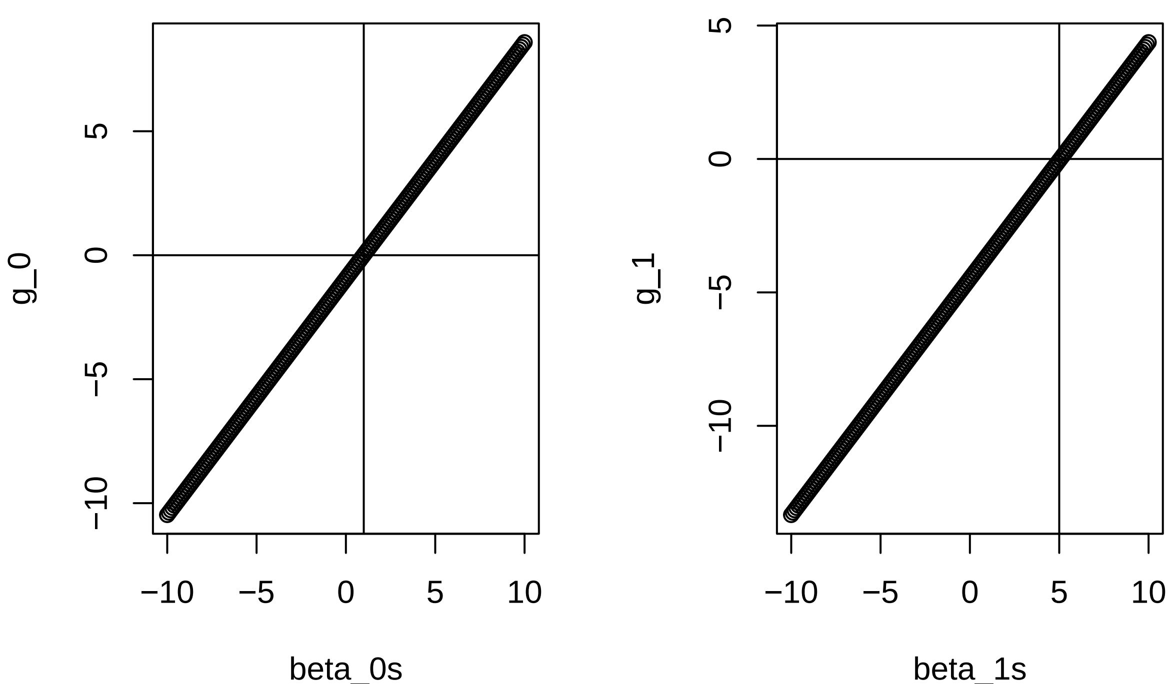
```
    count_same <- 0

    while(count_same < 10){

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

        #print(cost_start)

        #print(c(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)

        # Calculate new cost.
        cost_after <- ols_cost(x, y, beta_0, beta_1)

        #print(cost_after)

        #print(c(beta_0, beta_1))

        # Check cost relation.
        #if(cost_start < cost_after) {
            #return("Bad alpha, over shot minimum! Lower alpha and try again")
        if(cost_start == cost_after) {
            df_results <- rbind(df_results,
                                c(beta_0, beta_1, cost_after))
            count_same <- count_same + 1
        } else {
            df_results <- rbind(df_results,
                                c(beta_0, beta_1, cost_after))
        }
    }
    return(df_results)
}
```

## Test

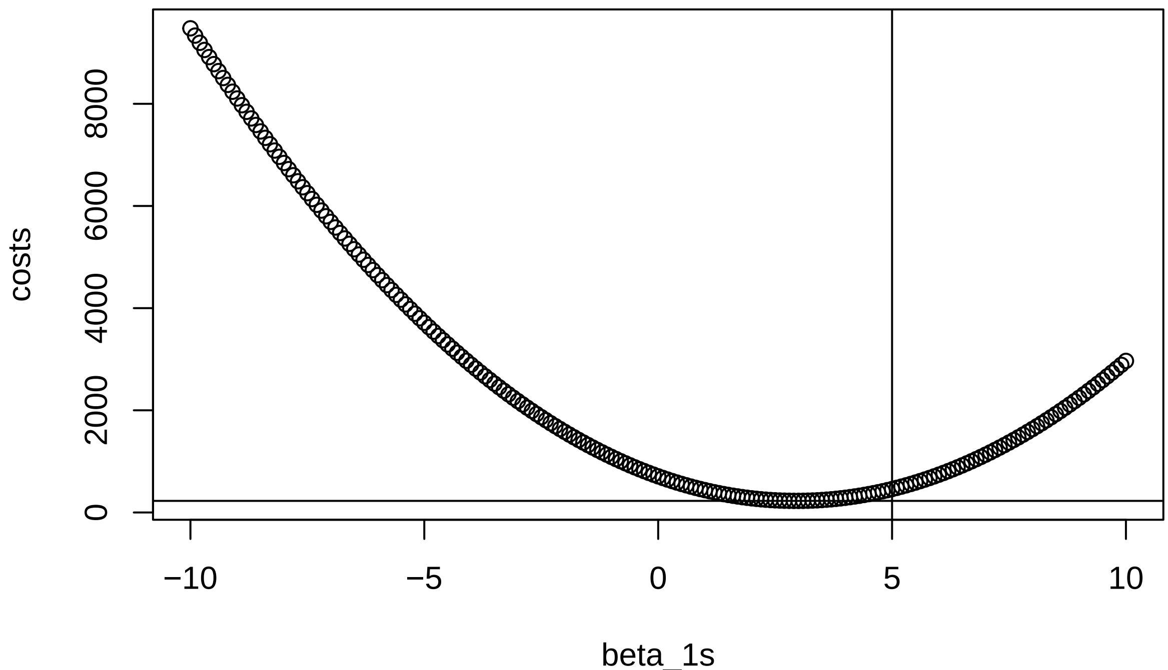
```
test_1 <- slr_gradient_desc(x, y, alpha = 0.01)
```

```
# gradient testing

# Summary statistic calculations.
x_bar <- mean(x)
y_bar <- mean(y)
xy_bar <- mean(x*y)
```

```r
x_sqbar <- mean(x^2)

# Betas to plot
beta_0s <- seq(from = -10, to = 10, by = .1)
beta_1s <- seq(from = -10, to = 10, by = .1)

g_0 <- beta_0s + (beta_1s * x_bar) - y_bar
g_1 <- (beta_0s * x_bar) + (beta_1s * x_sqbar) - xy_bar

par(mfrow = c(1,2))
plot(beta_0s, g_0)
abline(h = 0, v = 1)
plot(beta_1s, g_1)
abline(h = 0, v = 5)
```




```r
costs <- c()

for(i in seq(from = 1, to = length(beta_0s))){
    costs <- append(costs, ols_cost(x, y, beta_0s[i], beta_1s[i]))
}

plot(beta_1s, costs)
abline(h = min(costs), v = 5)
```




**Linear Model on Airbnb Data**