

Regression

Input: \mathbf{x} 's (covariates, features, independent, predictors, explanatory variables)

Output: y (outcome, response, dependent variable)

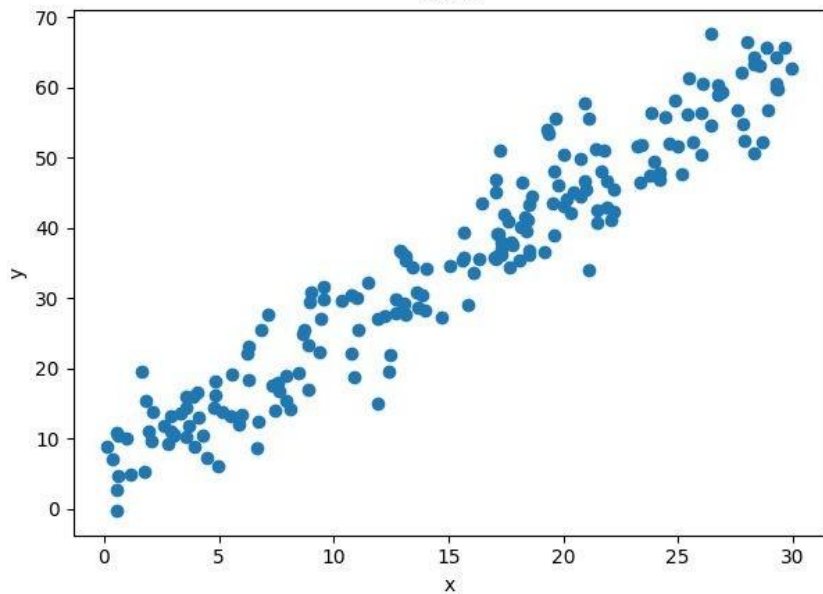
Goal: Find a regression function, $y \approx f(x, \beta)$

Simple Linear Regression: $y = \beta_0 + \beta_1 x$

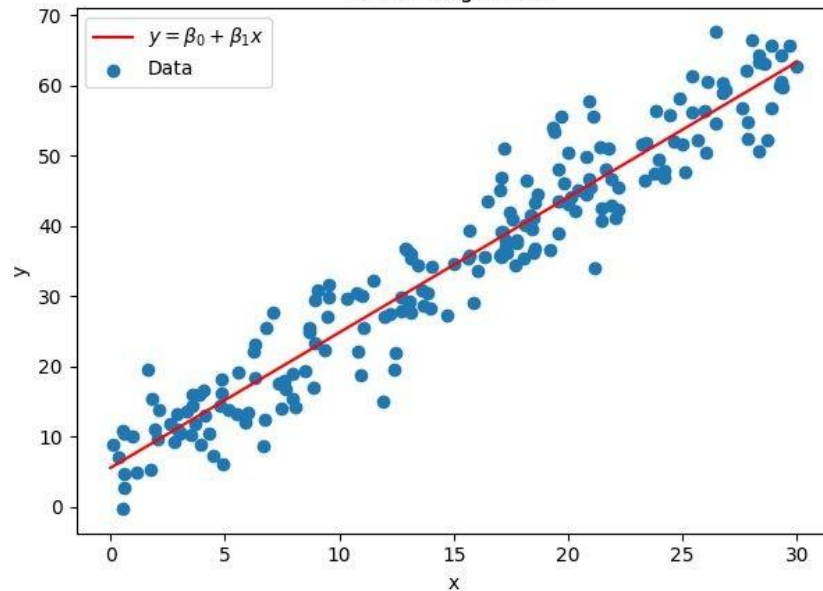
Multiple Linear Regression: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d$

Linear Regression

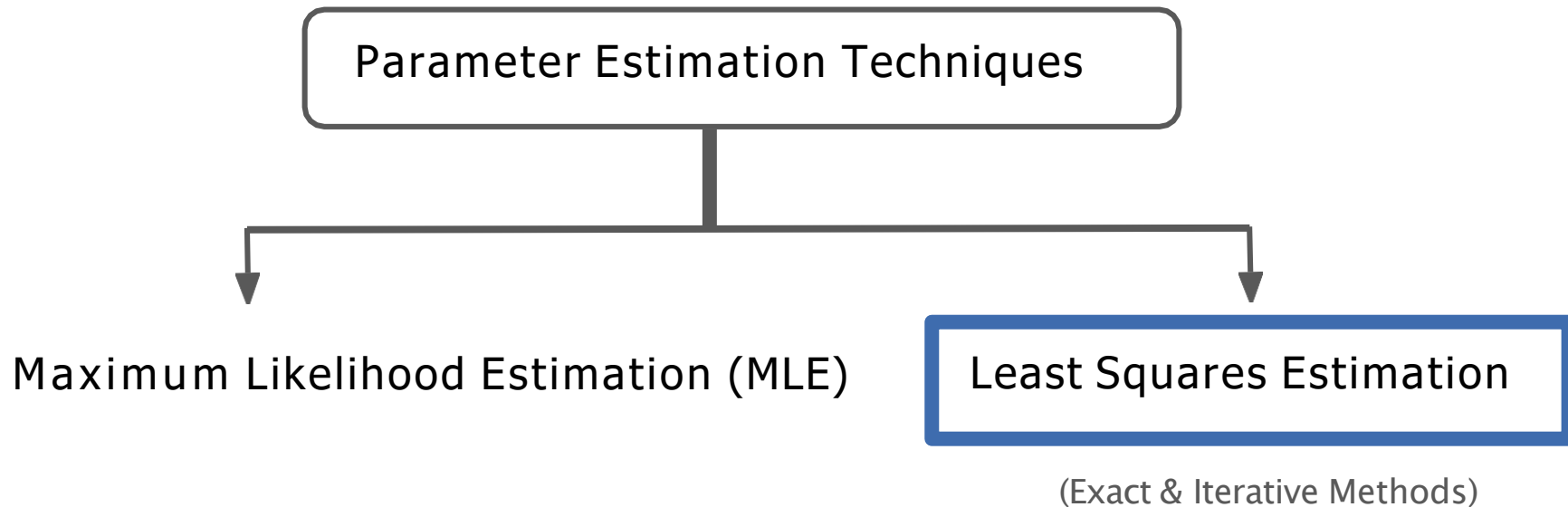
Data



Linear Regression



Parameter Estimation Technique for LR



Multiple Linear Regression

Multiple Linear regression model: $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d$

Goal: To find $\beta_0, \beta_1, \dots, \beta_d$

d  number of input features

Predicted Output value (\hat{y}): $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d$

Actual Output value (y): $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d + \epsilon$

Error or Residual (ϵ): $y - \hat{y}$

Multiple Linear Regression

For n set of observations:

$$\begin{aligned}y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_d x_{1d} + \epsilon_1 \\y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_d x_{2d} + \epsilon_2 \\y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \cdots + \beta_d x_{3d} + \epsilon_3 \\\vdots &\quad \quad \quad \vdots \\y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_d x_{nd} + \epsilon_n\end{aligned}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (1)$$

SSE for Multiple Linear Regression

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n \epsilon_i^2$$

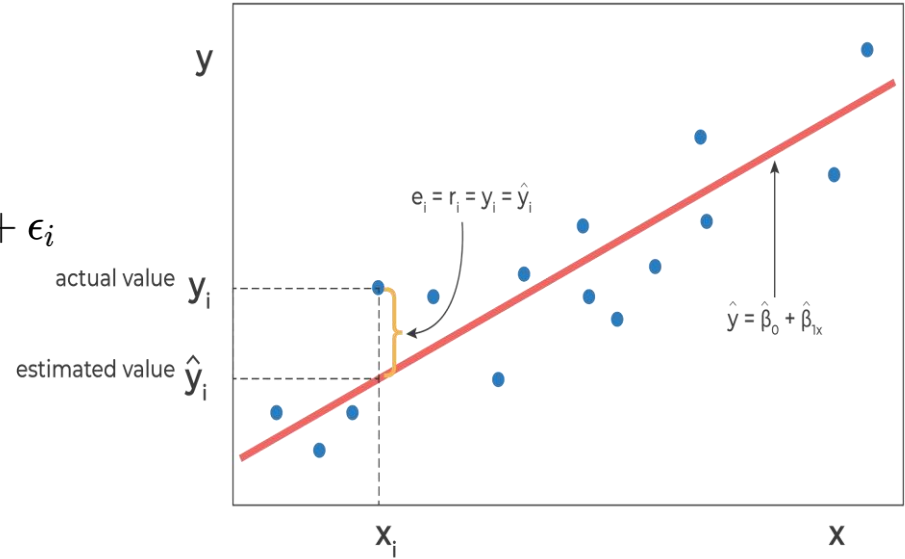
$$\epsilon^T \epsilon = [\epsilon_1 \quad \epsilon_2 \quad \dots \quad \epsilon_n] \cdot \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} = \epsilon_1^2 + \epsilon_2^2 + \dots + \epsilon_n^2 = \sum_{i=1}^n \epsilon_i^2$$

$$\text{SSE} = \epsilon^T \epsilon$$

Errors or Residuals

- Predicted Output value: $\hat{y}_i = \beta_0 + \beta_1 x_i$
- Observed Output value: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$

Error or Residual, $\epsilon_i = y_i - \hat{y}_i$



Sum of Squares of Errors (SSE)

SSE, also called Residual Sum of Squares(RSS) is always convex

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2$$

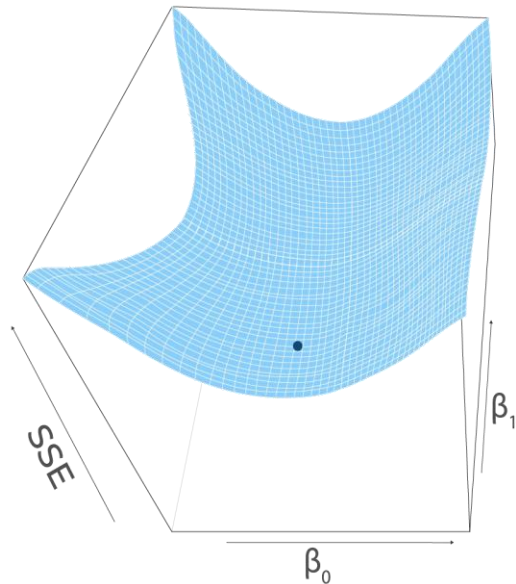


Figure: Plot of SSE

SSE for Multiple Linear Regression

From equation (1): $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$

So, $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$

$$\text{SSE} = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Solving for β

- Choose parameters $\beta_0, \beta_1, \dots, \beta_d$ such that **SSE is minimum**.
- Parameters at minimum point of SSE is obtained through setting:

First derivative of SSE w.r.t parameters = 0

- Take **partial derivative of SSE** with respect to column vector, β and **equate to 0**.

Taking Partial derivative w.r.t. beta

$$\frac{\partial \text{SSE}}{\partial \beta} = \frac{\partial}{\partial \beta} (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

$$= \frac{\partial}{\partial \beta} (\mathbf{y}^T \mathbf{y} + \beta^T \mathbf{X}^T \mathbf{X} \beta - 2\beta^T \mathbf{X}^T \mathbf{y})$$

$$= 2\mathbf{X}^T \mathbf{X} \beta - 2\mathbf{X}^T \mathbf{y}$$

$$\beta^T \beta = \beta^2$$

$$\frac{\partial \text{SSE}}{\partial \beta} = 2\mathbf{X}^T \mathbf{X} \beta - 2\mathbf{X}^T \mathbf{y} = 0$$

$$2\mathbf{X}^T \mathbf{X} \beta = 2\mathbf{X}^T \mathbf{y}$$

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Non Existence of $(\mathbf{X}^T \mathbf{X})^{-1}$ **and** $n \gg d$

- There should not be multicollinearity/dependence among the features.
- Number of Observations should be greater than the number of unknowns

Non-iterative methods find optimal parameters in a single iteration

Non-Iterative Method

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

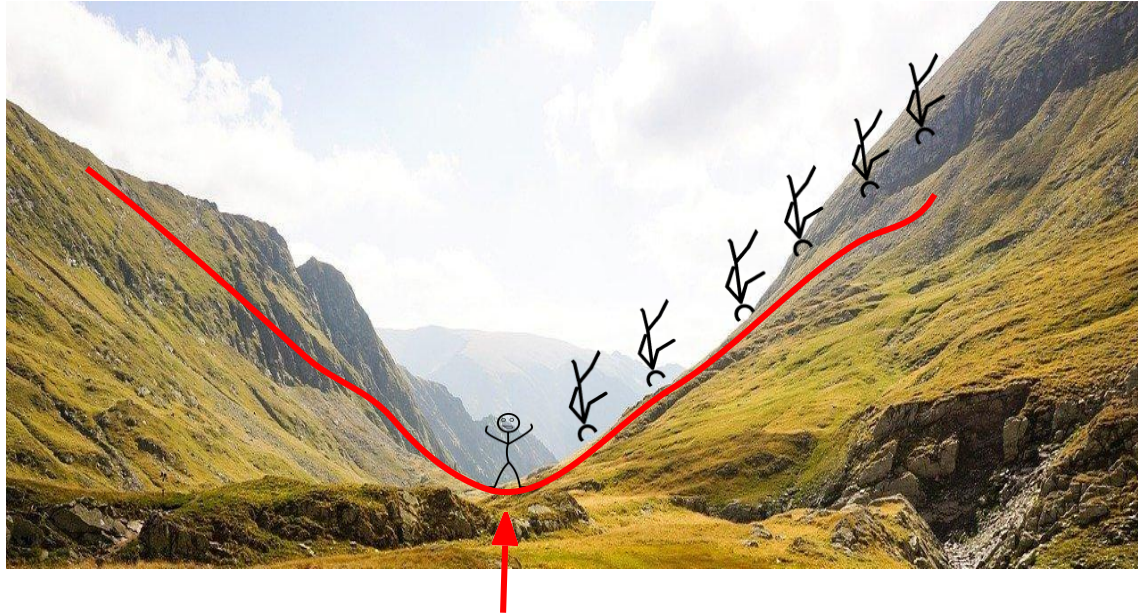
Single Iteration

Iterative Methods find optimal parameters using a number of iterations

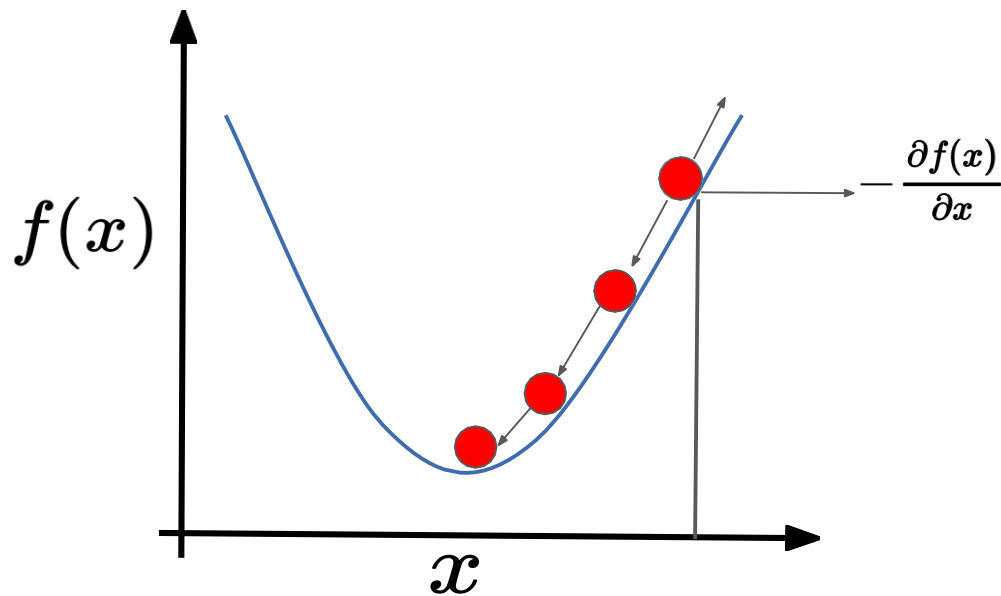
Iterative Method

- Random initialization of parameters
 - Sequence of approximations using an update rule
 - Gradient Descent is a **first order** iterative optimization approach
-

Gradient Descent is similar to getting down a hill...



Gradient Descent uses the gradient of the function to find the minimum point



Gradient Descent Algorithm

Step 1: Initialize the value of x randomly

Step 2: Calculate $\frac{\partial f(x)}{\partial x}$

Step 3: Update x as:

$$x := x - \alpha \frac{\partial f(x)}{\partial x}$$

Learning Rate

Step 4: Repeat steps 2, and 3 until convergence

Example

$$f(x) = x^2 + 3x - 5$$

$$x = 2.4$$

$$\alpha = 0.25$$

First iteration

$$\begin{aligned}x &:= x - \alpha \frac{\partial f(x)}{\partial x} \\&:= 2.4 - 0.25 \times 7.8 \\&:= 0.45\end{aligned}$$

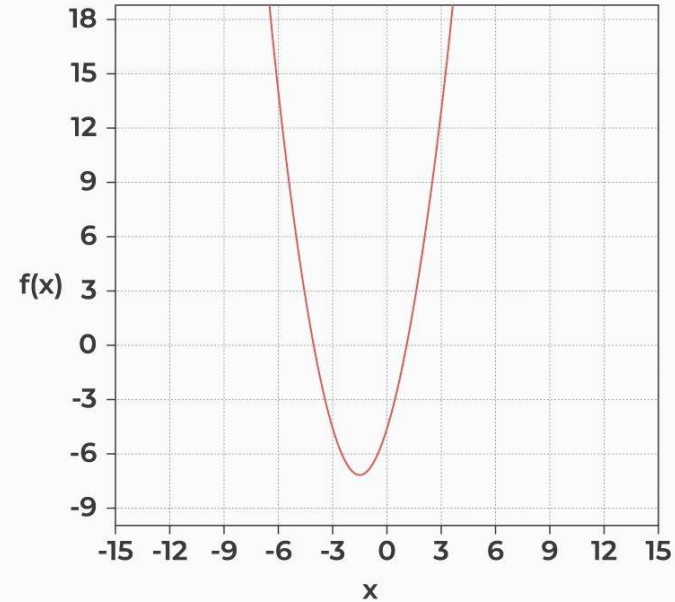
Second iteration

$$\begin{aligned}x &:= x - \alpha \frac{\partial f(x)}{\partial x} \\&:= 0.45 - 0.25 \times 3.9 \\&:= -0.525\end{aligned}$$

Example

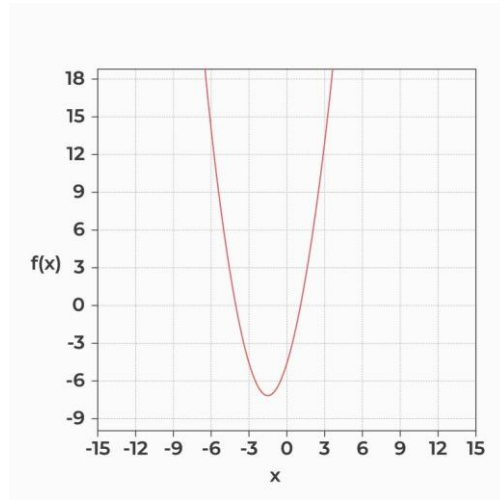
Optimal value of $x = -1.4999$

Min. of $f(x) = -7.25$



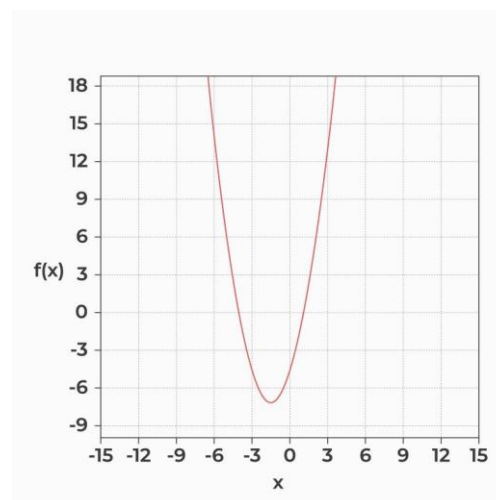
Learning Rate determines the size of the steps taken while moving towards the minimum of the function.

Too Small Learning Rate



Requires more iterations to converge

Too Large Learning Rate



Overshoots the minimum point in the function

Gradient Descent may get stuck in the local minima of non-convex functions

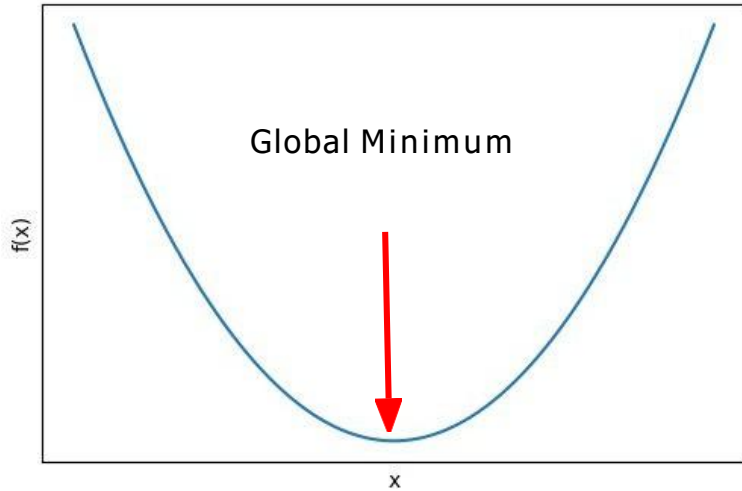


Fig. Convex Function

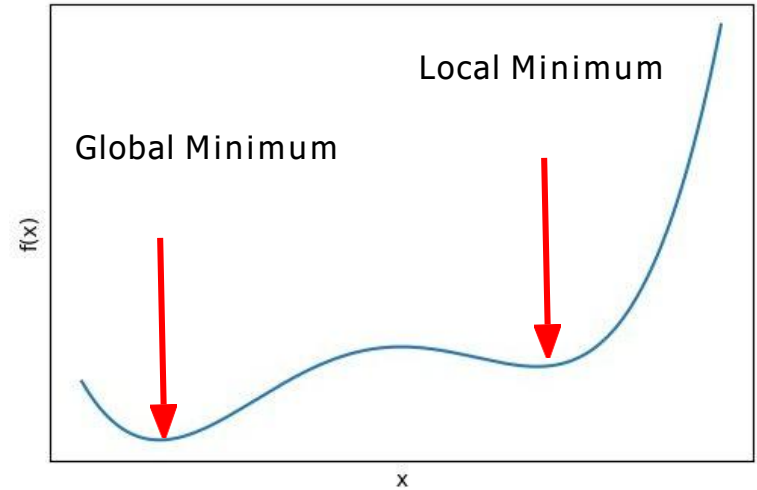


Fig. Non-convex Function

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{200} \end{bmatrix} = \begin{bmatrix} 1 & x_{1\ 1} & x_{1\ 2} & x_{1\ 3} \\ 1 & x_{2\ 1} & x_{2\ 2} & x_{2\ 3} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{200\ 1} & x_{200\ 2} & x_{200\ 3} \end{bmatrix} \times \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

$$\hat{y}_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}$$

Cost function for linear regression is the sum of squared error multiplied by $\frac{1}{2}$

$$\begin{aligned} J(\beta_0, \beta_1, \beta_2, \beta_3) &= \frac{1}{2} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \\ &= \frac{1}{2} \sum_{i=1}^n ((\beta_0 x_{i0} + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) - y_i)^2 \end{aligned}$$

Gradients can be calculated by finding the partial derivatives

$$\frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1, \beta_2, \beta_3) = \frac{\partial}{\partial \beta_1} \frac{1}{2} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^n \boxed{\frac{\partial}{\partial \beta_1} (\hat{y}_i - y_i)^2}$$

$$= \frac{1}{2} \sum_{i=1}^n \boxed{\frac{\partial (\hat{y}_i - y_i)^2}{\partial (\hat{y}_i - y_i)}} \times \frac{\partial (\hat{y}_i - y_i)}{\partial \beta_1}$$

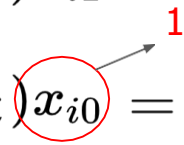
$$= \frac{1}{2} \sum_{i=1}^n \cancel{2} (\hat{y}_i - y_i) \times \boxed{\frac{\partial (\hat{y}_i - y_i)}{\partial \beta_1}}$$

$$= \sum_{i=1}^n (\hat{y}_i - y_i) \times \frac{\partial (\beta_0 x_{i0} + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) - y_i}{\partial \beta_1}$$

$$\boxed{= \sum_{i=1}^n (\hat{y}_i - y_i) x_{i1}}$$

Gradients w.r.t other parameters

$$\frac{\partial J}{\partial \beta_1} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{i1}$$

$$\frac{\partial J}{\partial \beta_0} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{i0} = \sum_{i=1}^n (\hat{y}_i - y_i)$$


$$\frac{\partial J}{\partial \beta_2} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{i2}$$

$$\frac{\partial J}{\partial \beta_3} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{i3}$$

$$\boxed{\frac{\partial J}{\partial \beta_j} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{ij}}$$

Gradient Descent uses the gradients to update the parameters

Repeat until convergence:

$$\beta_0 := \beta_0 - \alpha \frac{\partial J}{\partial \beta_0}$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial J}{\partial \beta_1}$$

$$\beta_2 := \beta_2 - \alpha \frac{\partial J}{\partial \beta_2}$$

$$\beta_3 := \beta_3 - \alpha \frac{\partial J}{\partial \beta_3}$$



$$\frac{\partial J}{\partial \beta_j} = \sum_{i=1}^n (\hat{y}_i - y_i) x_{ij}$$

Simultaneously updated

Gradient Descent on Advertisement Dataset

Learning rate = 0.0000003

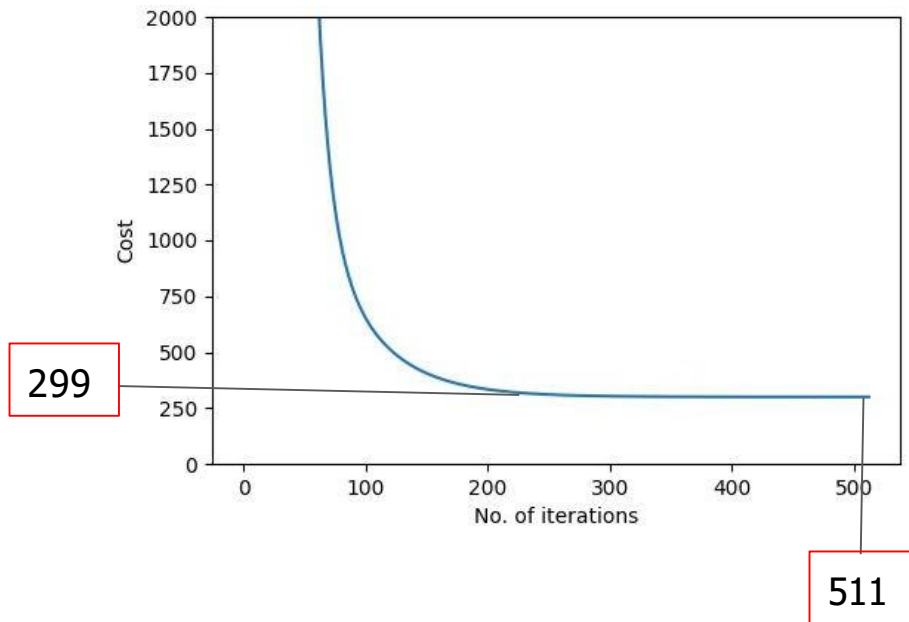
$$\beta_0 = 1.75$$

$$\beta_1 = 0.05$$

$$\beta_2 = 0.20$$

$$\beta_3 = 0.01$$

$$y = 1.75 + 0.05x_1 + 0.20x_2 + 0.01x_3$$



Comparison with OLS

OLS

- No need to choose any hyperparameter
- $n \gg d$, $(X^T X)^{-1}$
- Always gives exact solution
- $O(d^3)$

Gradient Descent

- Need to choose learning rate and number of iterations
- No such constraints
- May not always give exact solution
- $O(kd^2)$