Gradient Descent

- We have the cost function $J(\theta_0, \theta_1)$ and we want to minimize it.
- We start with some θ_0 and θ_1.
- Keep changing θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum.
- **Algorithm:**

  repeat until convergence

  {

  $$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j} \text{ for } j = 0 \text{ and } j = 1$$

  }

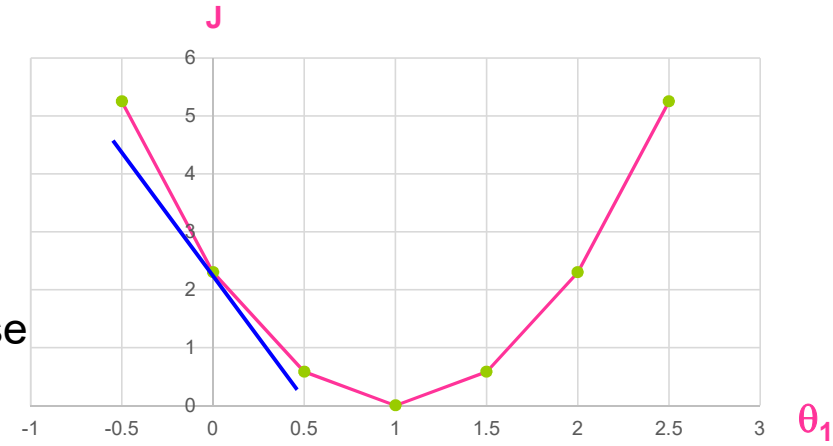
- Here α is the learning rate which controls how bigger step we take to reach the optimal solution.
- **NOTE:** We need to update θ_0 and θ_1 simultaneously.

Gradient Descent Intuition

- The learning rate (α) and the derivate term ($\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_j}$) work together to update θ_0 and θ_1.
- For better understanding consider the cost function J with one parameter θ_1.
  $\underset{\theta_1}{\text{Minimize}}\, J(\theta_1)$ where $\theta_1 \in R$

  We update $\theta_1 := \theta_1 - \alpha \frac{dJ(\theta_1)}{d\theta_1}$

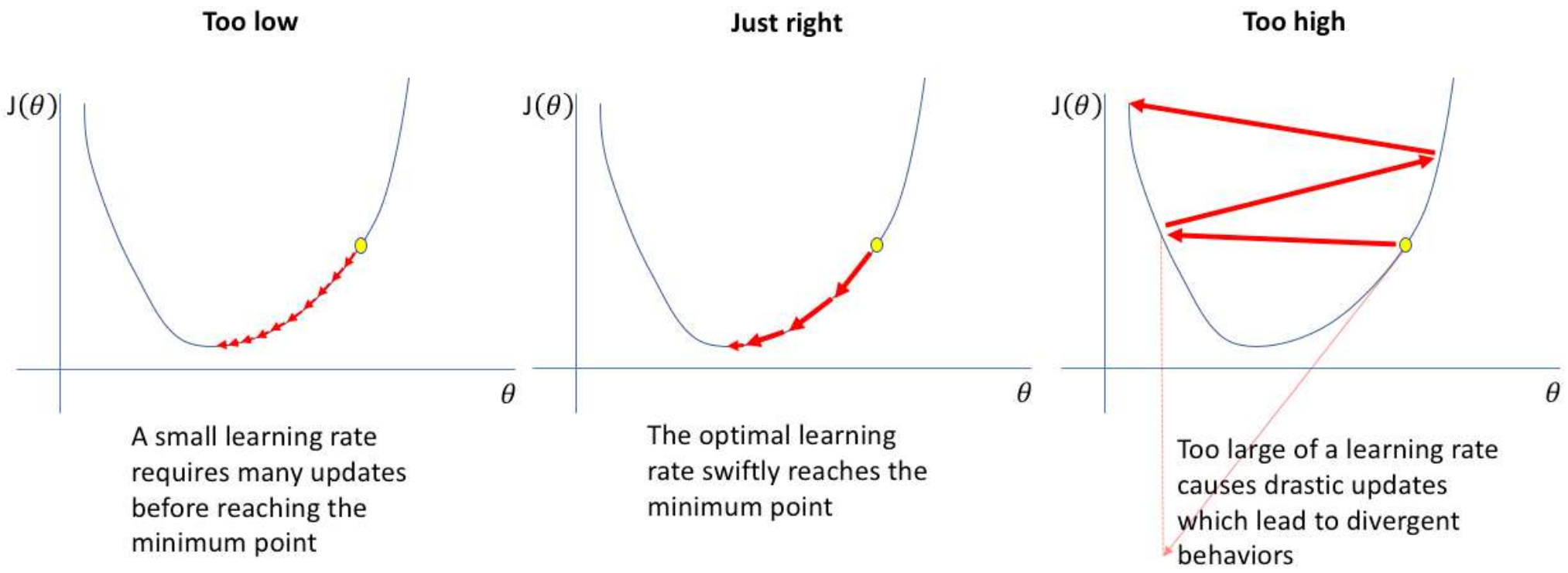
Slope of the tangent passing through the point.



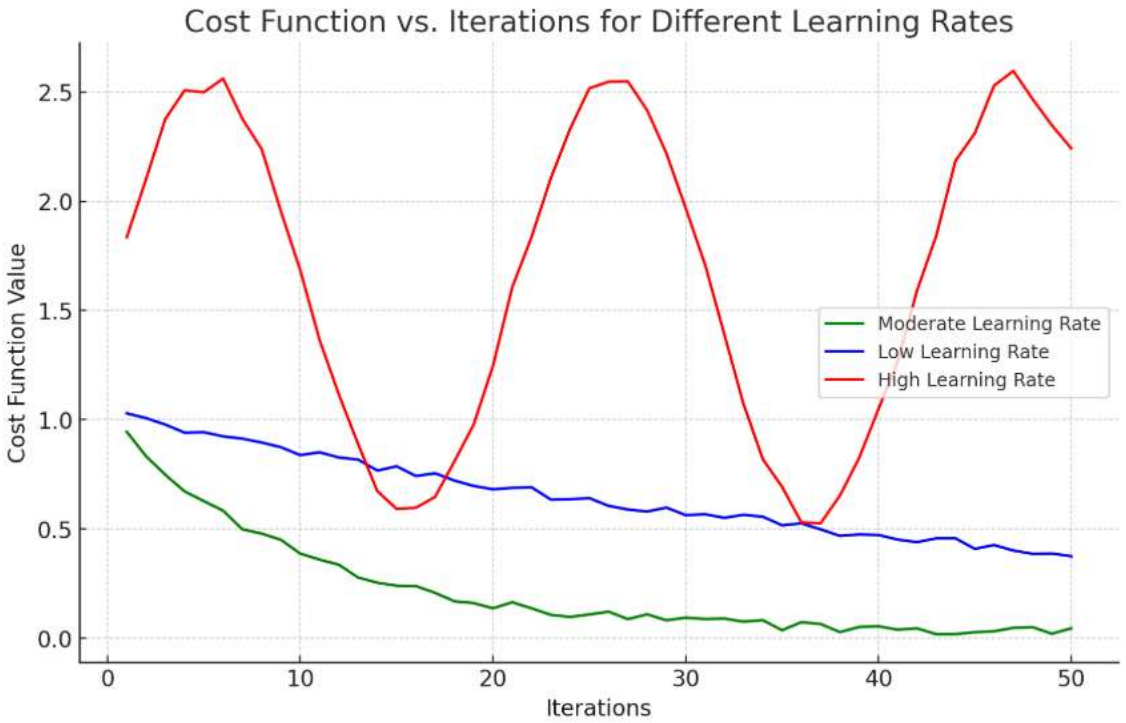
- If the right side of the tangent is pointing downward then slope is negative i.e. $\frac{dJ(\theta_1)}{d\theta_1} \leq 0$
- So when we update $\theta_1 := \theta_1 - \alpha(-ve\ value)$, we increase the value of θ_1 which is the right thing to do.
- Similarly, when we initialize θ_1 a point for which $\frac{dJ(\theta_1)}{d\theta_1} \geq 0$ we update $\theta_1 := \theta_1 - \alpha(+ve\ value)$, we decrease the value of θ_1.
- At local optimum $\frac{dJ(\theta_1)}{d\theta_1} = 0$.

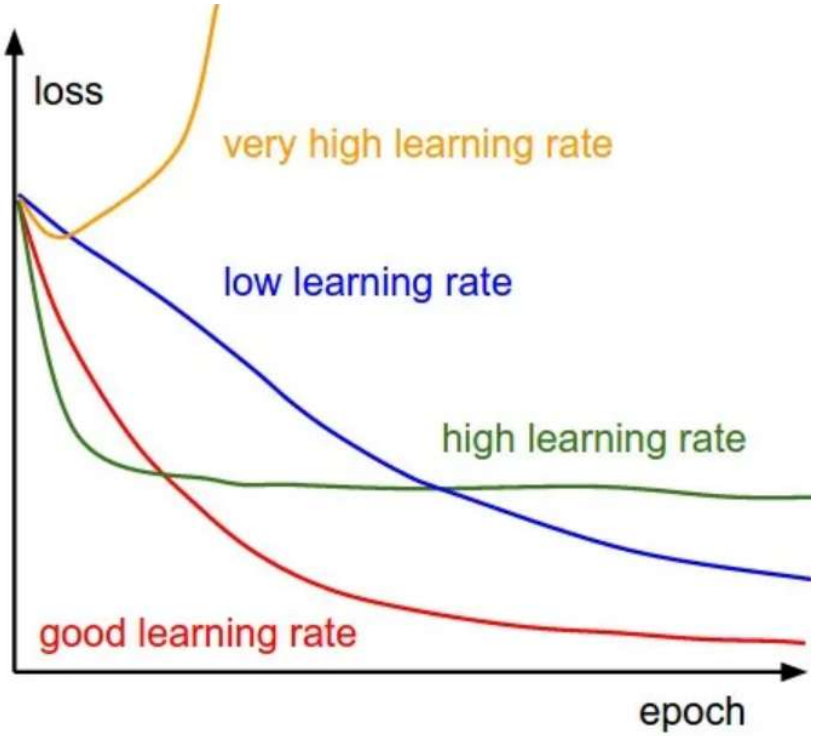
Gradient Descent Intuition

- If the learning rate is too small, gradient descent can be slow.
- If the learning rate is too large, gradient descent can overshoot the minimum. It may fail to converge or even diverge.



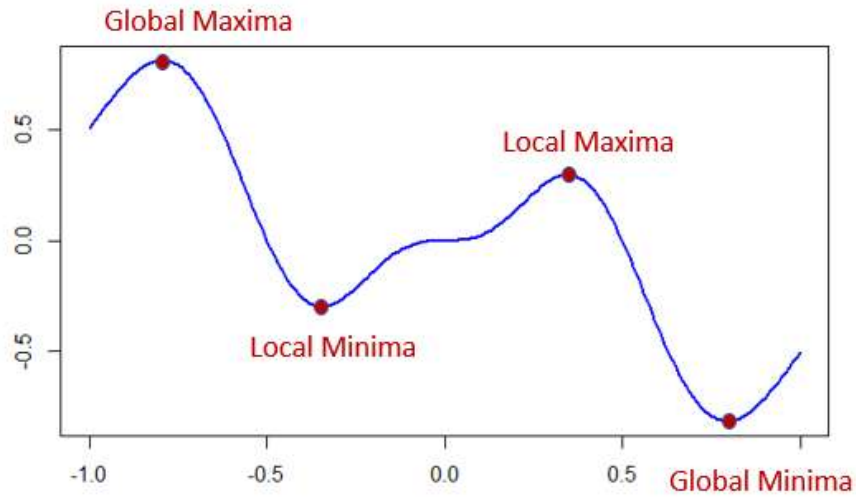
Effect of Learning Rate on Cost Function





Gradient Descent Intuition

- Gradient descent can converge to a local minimum, even with fixed learning rate.
- As we approach a local minimum gradient descent will automatically take smaller steps. So no need to decrease learning rate over time.



Calculation of Least Square gradient

- The cost function $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)})\right)^2$ where $\hat{y}^{(i)} = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$.
- Gradient of cost function, J with respect to parameters θ_0 and θ_1 are:
  - $\frac{\partial J}{\partial \theta_0} = \frac{1}{2m}\frac{\partial}{\partial \theta_0}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m} 2\left(y^{(i)} - \hat{y}^{(i)}\right)\left(-\frac{\partial \hat{y}^{(i)}}{\partial \theta_0}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)$

    $\left[\because \frac{\partial \hat{y}^{(i)}}{\partial \theta_0} = \frac{\partial[\theta_0 + \theta_1 x^{(i)}]}{\partial \theta_0} = 1\right]$
  - $\frac{\partial J}{\partial \theta_1} = \frac{1}{2m}\frac{\partial}{\partial \theta_1}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m} 2\left(y^{(i)} - \hat{y}^{(i)}\right)\left(-\frac{\partial \hat{y}^{(i)}}{\partial \theta_1}\right)$

    $= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)x^{(i)}\left[\because \frac{\partial \hat{y}^{(i)}}{\partial \theta_1} = \frac{\partial[\theta_0 + \theta_1 x^{(i)}]}{\partial \theta_1} = x^{(i)}\right]$
- Combining, $\boxed{\frac{\partial J}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)x_j{}^{(i)}}$ where $x_j \equiv x^j$ i.e. $x_0 \equiv x^0 = 1$, $x_1 \equiv x^1 = x$

Steps of Linear Regression Procedure

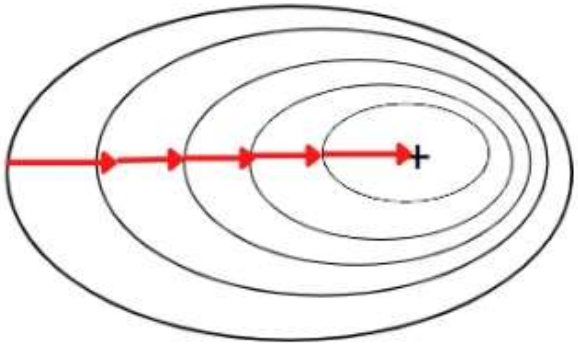
1. Decide on α, ε and stopping criterion.
2. Make an initial guess for the weight vector $\theta = \theta^{(0)}$.
3. Calculate $\theta^{(k+1)} = \theta^{(k)} - \frac{\alpha}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)x^{(i)}$
4. Calculate the stopping criterion
   a) If condition satisfied,
   b) If not satisfied go to Step 3

Types of Gradient Descent

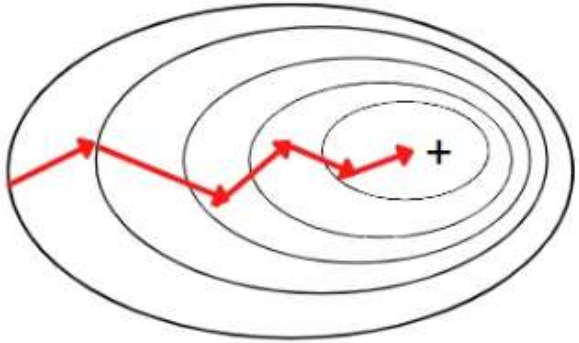
1. Batch Gradient Descent:
   - Makes a parameter update i.e calculates gradient of cost function, using the entire training data set.
   - No online update is possible.
2. Stochastic Gradient Descent:
   - Parameter updates are done for every training sample.
   - Online learning is possible
   - Causes large oscillations in objective function due to frequent updates.
3. Mini-batch Gradient Descent:
   - Combination of batch gradient descent and stochastic gradient descent.
   - Performs update for a mini-batch of training data.
   - It uses random subsets (mini-batches). The randomness introduces noise into the optimization process, which can help the algorithm escape local optima.
   - Use smaller mini-batch sizes for more randomness.

Types of Gradient Descent

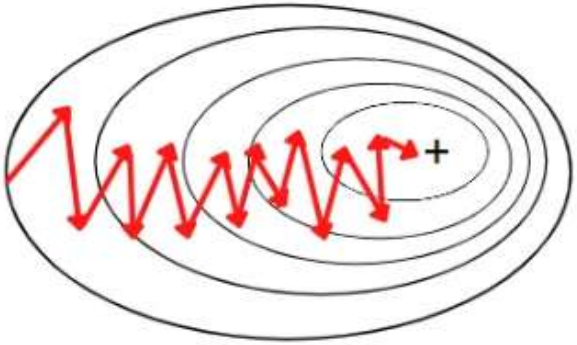
Batch Gradient Descent



Mini-Batch Gradient Descent



Stochastic Gradient Descent



Error Metrics / Goodness of Fit

- To evaluate the performance of the regression model, we use error metrics to quantify how well the model's predictions match the actual data.
- **Goodness of Fit** in machine learning refers to how well a model's predictions align with the observed data. It assesses the quality of the model in representing the underlying data-generating process.
- Understanding these metrics is crucial for assessing the quality of a regression model and making improvements.
- Common Goodness of Fit Measures:
  - Mean Absolute Error (MAE)
  - Sum Square Error (SSE)
  - Mean Squared Error (MSE)
  - Root Mean Squared Error (RMSE)
  - Total Sum of Squares (TSS)
  - Sum Squared Regression (SSR)
  - Mean Absolute Percentage Error (MAPE)
  - R-Squared (Coefficient of Determination)
  - Adjusted R-Squared

Error Metrics / Goodness of Fit

1. **Mean Absolute Error (MAE)**: Measures the average of absolute difference between actual and predicted values.
   - $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
   - Lower value indicates a better fit.
2. **Sum Squared Error (SSE) / Residual Sum of Squares (RSS):** Measures the sum squared difference between predicted and actual values. Amount of error found by the model.
   - $SSE/RSS = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
   - Lower value indicates a better fit.
3. **Mean Squared Error (MSE) :** Measures the average squared difference between predicted and actual values.
   - $MSE = \frac{SSE}{N} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
   - Lower value indicates a better fit.

Error Metrics / Goodness of Fit

4. **Root Mean Squared Error (RMSE):** Measures the average squared difference between predicted and actual values.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Lower value indicates a better fit

5. **Total Sum of Squares (TSS):** Amount of variance present in the data.

$$\text{TSS} = \sum_{i=1}^{N}(y_i - \bar{y})^2 \; where \; \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Lower value indicates less variance in the data

6. **Sum Squared Regression (SSR):** Amount of variance captured by the model.

$$\text{SSR} = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2 \; where \; \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Lower value indicates less variance in the predicted output.

7. **Mean Absolute Percentage Error (MAPE):** Average of absolute percentage error between actual and predicted values.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Lower value indicates a better fit

Error Metrics / Goodness of Fit

8. **R-Squared (Coefficient of Determination):** Measures the proportion of variance in the dependent variable Y that is predictable from the independent variable X.

   - $R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$
   - $R^2 \epsilon [0,1]$, $R^2 = 1$: Perfect Fit, $R^2 = 0$: No explanatory power
   - Lower value indicates a better fit.

9. **Adjusted R-Squared:** Adjust the R^2 value based on the number of predictors in the model. It accounts for the number of predictors in the model, penalizing unnecessary complexity.

$$Adjusted\ R^2 = 1 - \frac{\frac{RSS}{N-k-1}}{\frac{TSS}{N-1}} = 1 - \frac{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N-k-1}}{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N-1}}$$

where N: number of observations, k: number of predictors or features.

   - Higher value indicates a better fit.

Multiple Linear Regression

- Given a data set $D = \{(x_i, y_i)\}, i = 1, 2, 3, \ldots, N$ of N patterns where $x_i \in R^p$, $y_i \in R$ and a new predictor vector $x \in R^p$, $x \notin D$. Determine the response $y \in R$ is associated with $x \in R^p$.
- Solving a regression problem involves obtaining a relation between the response Y and predictors i.e., determine a function $f(X_1, X_2, \ldots, X_p)$ such that $Y = f(X) + \epsilon$ (ϵ is called a random error such that $E(\epsilon) = 0$) based on the given set D of patterns.

- Goal: Determine $\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$ to satisfy the following equations:

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \ldots \beta_p x_{1p} \\
y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \ldots \beta_p x_{2p} \\
&\vdots \\
y_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip} \\
&\vdots \\
y_N &= \beta_0 + \beta_1 x_{N1} + \beta_2 x_{N2} + \ldots \beta_p x_{Np}
\end{aligned}
$$

---------- (1)

N equations and (p+1) unknowns.

Usually N > p+1

Multiple Linear Regression

- We will use Least Square Method for determination of β.
- We have $\hat{y}_i = f(x_i)$, where $\hat{y}_i$ denotes the computed value from the model for input x_i.
- Residual error = $y_i - \hat{y}_i$.
- Residual Sum of Squares (RSS) is defined as, $RSS(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

Fact from Optimization

- An univariate function $f(x)$ is minimized for some $x = x^*$ if $\frac{df}{dx}\Big]_{x=x^*} = 0$ and $\frac{d^2f}{dx^2}\Big]_{x=x^*} > 0$.
- A given multivariate function $f(x_1, x_2, \ldots, x_n)$ is minimized for some $x = x^*$ if $\nabla f]_{x=x^*} = 0$ and Hessian of f at $x = x^*$ is positive definite, where

$$\text{Gradient of } f,\ \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \text{ and Hessian of } f = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \frac{\partial^2 f}{\partial x_2 \partial 3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial {x_3}^2} & \cdots & \frac{\partial^2 f}{\partial x_3 \partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \frac{\partial^2 f}{\partial x_n \partial x_3} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix}$$

Example of Multivariate Optimization

$$\min_{x_1, x_2} \quad x_1 + 2x_2 + 4x_1^2 - x_1 x_2 + 2x_2^2$$

First order condition

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 + 8x_1 - x_2 \\ 2 - x_1 + 4x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

solving

$$\begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} -0.19 \\ -0.54 \end{bmatrix}$$

Second order condition

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 8 & -1 \\ -1 & 4 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 3.76 \\ 8.23 \end{bmatrix}$$

$$\begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} -0.19 \\ -0.54 \end{bmatrix}$$

New Goal

- Determine β so as to minimise $RSS(\beta) = \sum_{i=1}^{N}[\, y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip})]^2$ is a function of $f(\beta_0, \beta_1, \ldots, \beta_p)$
- Gradient of f, $\nabla f = \begin{pmatrix} \frac{\partial f}{\partial \beta_0} \\ \frac{\partial f}{\partial \beta_1} \\ \frac{\partial f}{\partial \beta_2} \\ \vdots \\ \frac{\partial f}{\partial \beta_p} \end{pmatrix}$ and Hessian of f = $\begin{bmatrix} \frac{\partial^2 f}{\partial \beta_0^2} & \frac{\partial^2 f}{\partial \beta_0 \partial \beta_1} & \frac{\partial^2 f}{\partial \beta_0 \partial \beta_2} & \cdots & \frac{\partial^2 f}{\partial \beta_0 \partial \beta_p} \\ \frac{\partial^2 f}{\partial \beta_1 \partial \beta_0} & \frac{\partial^2 f}{\partial \beta_1^2} & \frac{\partial^2 f}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial^2 f}{\partial \beta_1 \partial \beta_p} \\ \frac{\partial^2 f}{\partial \beta_2 \partial \beta_0} & \frac{\partial^2 f}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 f}{\partial \beta_2^2} & \cdots & \frac{\partial^2 f}{\partial \beta_2 \partial \beta_p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial \beta_n \partial \beta_0} & \frac{\partial^2 f}{\partial \beta_n \partial \beta_1} & \frac{\partial^2 f}{\partial \beta_n \partial \beta_2} & \cdots & \frac{\partial^2 f}{\partial \beta_p^2} \end{bmatrix}$
- $RSS(\beta)$ is minimized for some $\beta = \hat{\beta}$ if $\nabla f]_{\beta=\hat{\beta}} = 0$ and Hessian of f at $\beta = \hat{\beta}$ is positive definite.

Matrix Calculus

1. If y is a scalar and x is a column vector, i.e. $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ then $\nabla_x y = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$

2. If x, y are column vectors, i.e. $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ and $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$ then $\nabla_x y = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$

3. If $y = f(x)$ and $x = \emptyset(z)$, where x, y, z are scalars then $\frac{dy}{dz} = \frac{dy}{dx} . \frac{dx}{dz} = \frac{dx}{dz} . \frac{dy}{dx}$

Matrix Calculus Formulae

If x is a column vector and A is a Matrix then

y	$\frac{\partial y}{\partial x}$
Ax	$\frac{\partial (Ax)}{\partial x} = A^T$
$x^T A$	$\frac{\partial (x^T A)}{\partial x} = A$
$x^T x$	$\frac{\partial (x^T x)}{\partial x} = 2x$
$x^T Ax$	$\frac{\partial (x^T Ax)}{\partial x} = Ax + A^T x = 2Ax$ if A is symmetric
$x^T \alpha = \alpha^T x$	α is a column vector independent of x

Multiple Linear Regression

- Let $X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}_{N\times(p+1)}$,$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N\times 1}$, $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}_{(p+1)\times 1}$
- Then $RSS(\beta) = \sum_{i=1}^{N}[\, y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip})]^2 = (y - X\beta)^T(y - X\beta)$
- Hence, $\frac{\partial RS\ (\beta)}{\partial \beta} = \frac{\partial (z^T z)}{\partial \beta} = \frac{\partial (z^T z)}{\partial z}\frac{\partial z}{\partial \beta} = \frac{\partial z}{\partial \beta}\frac{\partial (z^T z)}{\partial z} = \frac{\partial (y - X\ )}{\partial \beta} 2z = \left[\frac{\partial y}{\partial \beta} - \frac{\partial (X\beta)}{\partial \beta}\right] 2z = \left[0 - \frac{\partial (X\beta)}{\partial \beta}\right] 2z = [-X^T]2z = [-X^T]2(y - X\beta) = -2X^T(y - X\beta)$, where

$$z = y - X\beta$$

- Since $\frac{\partial RSS(\beta)}{\partial \beta} = 0 \Longrightarrow -2X^T(y - X\beta) = 0 \Longrightarrow X^T(y - X\beta) = 0 \Longrightarrow X^T y - X^T X\beta = 0$

$\Longrightarrow X^T X\beta = X^T y \Longrightarrow \boldsymbol{\beta = (X^T X)^{-1} X^T y = \hat{\beta}}$ (Say)

Multiple Linear Regression

Hesian Matrix of $RSS(\beta) =$

$$\begin{bmatrix} \frac{\partial}{\partial\beta_0}\left(\frac{\partial RSS}{\partial\beta_0}\right) & \frac{\partial}{\partial\beta_0}\left(\frac{\partial RSS}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_0}\left(\frac{\partial RSS}{\partial\beta_2}\right) & \cdots & \frac{\partial}{\partial\beta_0}\left(\frac{\partial RSS}{\partial\beta_p}\right) \\ \frac{\partial}{\partial\beta_1}\left(\frac{\partial RSS}{\partial\beta_0}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial RSS}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial RSS}{\partial\beta_2}\right) & \cdots & \frac{\partial}{\partial\beta_1}\left(\frac{\partial RSS}{\partial\beta_p}\right) \\ \frac{\partial}{\partial\beta_2}\left(\frac{\partial RSS}{\partial\beta_0}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial RSS}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial RSS}{\partial\beta_2}\right) & \cdots & \frac{\partial}{\partial\beta_2}\left(\frac{\partial RSS}{\partial\beta_p}\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial\beta_N}\left(\frac{\partial RSS}{\partial\beta_0}\right) & \frac{\partial}{\partial\beta_N}\left(\frac{\partial RSS}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_N}\left(\frac{\partial RSS}{\partial\beta_2}\right) & \cdots & \frac{\partial}{\partial\beta_N}\left(\frac{\partial RSS}{\partial\beta_p}\right) \end{bmatrix} = \frac{\partial}{\partial\beta}\left(\frac{\partial RSS(\beta)}{\partial\beta}\right) = \frac{\partial}{\partial\beta}[-2X^T(y - X\beta)]$$

$$= -2\frac{\partial}{\partial\beta}[X^T(y - X\beta)] = -2\frac{\partial}{\partial\beta}[X^Ty - X^TX\beta] = -2\left[0 - \frac{\partial}{\partial\beta}(X^TX\beta)\right] = 2\frac{\partial}{\partial\beta}(X^TX\beta) = 2(X^TX)^T = 2X^TX$$

NOTE:

1. For any matrix X we can prove that X^TX is a square matrix and symmetric matrix.
2. Any symmetric matrix S is positive definite if and only if there is a matrix X with independent columns such that $S = X^TX$
3. X^TX is invertible if columns of X has full column rank (columns are independent)

Prediction

- For a new $x_0 = R^p$ the response $y_0 = \hat{\beta}_0 + \hat{\beta}_1 x_{01} + \hat{\beta}_2 x_{02} + \ldots + \hat{\beta}_p x_{0p} = \hat{\beta}^T \tilde{x}_0$ where

$$\tilde{x}_0 = \begin{bmatrix} 1 \\ x_0 \end{bmatrix} = \begin{bmatrix} 1 \\ x_{01} \\ x_{02} \\ \vdots \\ x_{0p} \end{bmatrix}$$

Multiple Linear Regression: Algorithm

Input: $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \epsilon R^p$, $y_i \epsilon R$ and a new predictor vector $x_0 \epsilon R^p$

Output: y_0

1. Create the matrix X and the vector y
2. Compute $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y})$
3. Create the $\tilde{x}_0 = \begin{bmatrix} 1 \\ x_0 \end{bmatrix} = \begin{bmatrix} 1 \\ x_{01} \\ x_{02} \\ \vdots \\ x_{0p} \end{bmatrix}$
4. Return $\hat{\beta}^T . \tilde{x}_0$

NOTE:

- $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y})$ if the columns of X are linearly independent
- Response Y is linearly related with the predictors

Example of Multiple Linear Regression using Normal Equation

Question:

Consider the following dataset:

X_1	X_2	Y
1	1	1
2	3	7
3	4	9
4	6	13

$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$. Find the optimal values of β using Normal Equation (Least Square Method).

Solution:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 6 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 7 \\ 9 \\ 13 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 4 & 10 & 14 \\ 10 & 30 & 43 \\ 14 & 73 & 62 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} 2.75 & -4.5 & 2.5 \\ -4.5 & 13 & -8 \\ 2.5 & -8 & -5 \end{bmatrix}$$

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

$$= \begin{bmatrix} 2.75 & -4.5 & 2.5 \\ -4.5 & 13 & -8 \\ 2.5 & -8 & -5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 7 \\ 9 \\ 13 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ 3 \end{bmatrix}$$

$$\hat{y} = X\hat{\beta} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 6 \end{bmatrix} \begin{bmatrix} -0.5 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 6.5 \\ 8.5 \\ 13.5 \end{bmatrix}$$

Gradient Descent for Multiple Linear Regression

- We know $\beta_j := \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$ for $j = 0, 1, 2, \ldots, p$
- $J(\beta) = \frac{1}{2N} \sum_{i=1}^{N} [\, y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip})]^2$
- $\frac{\partial J(\beta)}{\partial \beta_j} = -\frac{1}{N} \sum_{i=1}^{N} (\, y_i - \hat{y}_i) x_{ij}$ for $j = 0$, $x_{ij} = 1$
- Algorithm:
  1. Initialize the coefficient $\beta_0, \beta_1, \ldots, \beta_p$.
  2. Calculate the predicted values $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ using the current coefficients.
  3. Compute the cost function $J(\beta)$.
  4. Make the update $\beta := \beta - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$
  5. Repeat steps 2-4 until convergence.

Direct Method (Normal Equation) vs Gradient Descent

Aspect	Direct Method (Normal Equation)	Gradient Descent
Efficiency for Small Datasets	Fast and efficient.	May be slower due to iterations.
Efficiency for Large Datasets	Computationally expensive due to matrix inversion.	More efficient as it avoids matrix inversion.
Regularization	Harder to implement.	Easily integrates regularization.
Scalability	Poor for high-dimensional data.	Scales well with large datasets.
Implementation	Requires solving a closed-form equation.	Requires iterative optimization.
Sparsity	Not memory-efficient for sparse datasets.	Works well with sparse data.

Limitation of Multiple Linear Regression

- If the data are not linearly distributed, then the prediction accuracy of the Multiple Linear Regression is low.
- If some of the features are highly correlated, then the matrix X may become computationally singular and hence $\hat{\beta}$ cannot be computed. If two features are highly correlated, then remove one of them but not both.
- When the number of features (p) is large, linear regression fit does not predict well even when the data are not non-linear.
- To overcome these limitations one of the approaches is shrinking the coefficient (β's) of the model or setting some of the β's equal to zero.

Polynomial Regression

- Polynomial regression is used when the relationship between the independent variable (input) and dependent variable (output) is non-linear. Unlike linear regression which fits a straight line, it fits a polynomial equation to capture the curve in the data.
- Polynomial regression is an extension of linear regression where higher-degree terms are added to model non-linear relationships.
- The general form of the equation for a polynomial regression of degree n is:

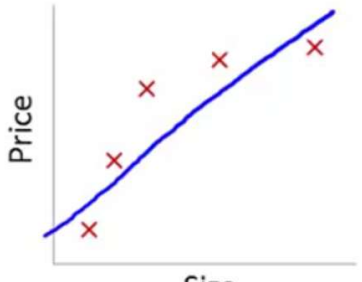
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \epsilon$$

- **Example:**

Years of Experience	Salary (in dollars)
1	50,000
2	55,000
3	65,000
4	80,000
5	1,10,000
6	1,50,000
7	2,00,000

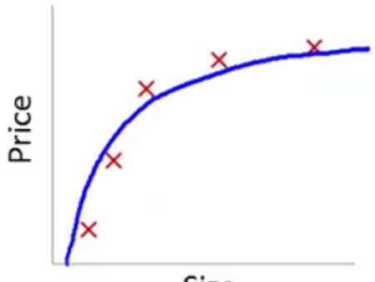
Polynomial Regression

- **Choosing the right polynomial degree n is important**: a higher degree may fit the data more closely but it can lead to overfitting.
- The degree should be selected based on the complexity of the data. Once the model is trained, it can be used to make predictions on new data, capturing non-linear relationships and providing a more accurate model for real-world applications.



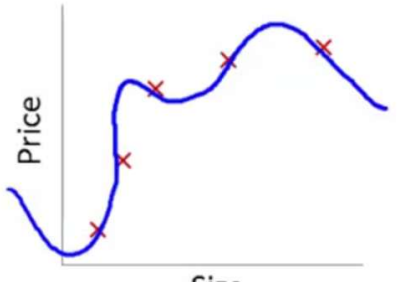
$\theta_0 + \theta_1 x$

Degree: 1



$\theta_0 + \theta_1 x + \theta_2 x^2$

Degree: 2



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Degree: 4

How to Find the Optimal Degree

- **Understand the Goal:** You're looking for the lowest 'n' (degree) that captures the true pattern in your data without fitting the random noise (overfitting).
- **Start Simple:** Begin with a low degree (like n=1, which is linear, or n=2, quadratic) and gradually increase it.
- **Split Data:** Divide your dataset into training and validation/test sets.
- **Train Models:** Fit polynomial models of increasing degrees (e.g., n=1, 2, 3, 4, 5...) to the *training* data.
- **Evaluate Performance:** Calculate the Root Mean Squared Error (RMSE) or other metrics for each model on the *validation* set.
- **Identify the Sweet Spot:** The optimal 'n' is usually found at the point where validation error (RMSE) is minimized; beyond this point, the error starts increasing (overfitting).
- **Use Tools:** Utilize functions in software like scikit-learn (Python), MATLAB's polyfit, or Excel's matrix functions to automate fitting and evaluation.

Feature Scaling

- **Feature scaling** is one of the most important data preprocessing step in machine learning.
- Algorithms that compute the distance between the features are biased towards numerically larger values if the data is not scaled.
- In Machine Learning, **Normalization** and **Standardization** are two important techniques used to scale data.
- Both methods are employed to make features comparable and improve the performance of algorithms that are sensitive to the scale of the input features.

Normalization

- Normalization scales the data to a fixed range, typically [0, 1] or [-1, 1], without altering the shape of the distribution of the data.
- **Formula for Normalization:**

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x is the original value

x_{min} and x_{max} are the minimum and maximum values in the dataset

- Normalization is useful when you know the data follows a uniform distribution.
- It ensures all features have the same range, which is particularly important for distance-based algorithms like **K-Nearest Neighbors (KNN)** and **Support Vector Machines (SVM)**.
- Works best when outliers are not present in the data, as they can disproportionately affect the scaling.
- **Example:**

Original data: [10, 20, 30, 40, 50]
Normalized data (range 0-1): [0, 0.25, 0.5, 0.75, 1]