Today, Agenda!

1. Dataset splitting
2. Overfit and underfit concepts
3. Gradient descent
4. R2 Value
5. Concept of multi-variable LR

Overfitting and Underfitting

Overfitting and Underfitting

Overview

Training Performance: Low (Gorib)
Testing Performance: Low (Gorib)

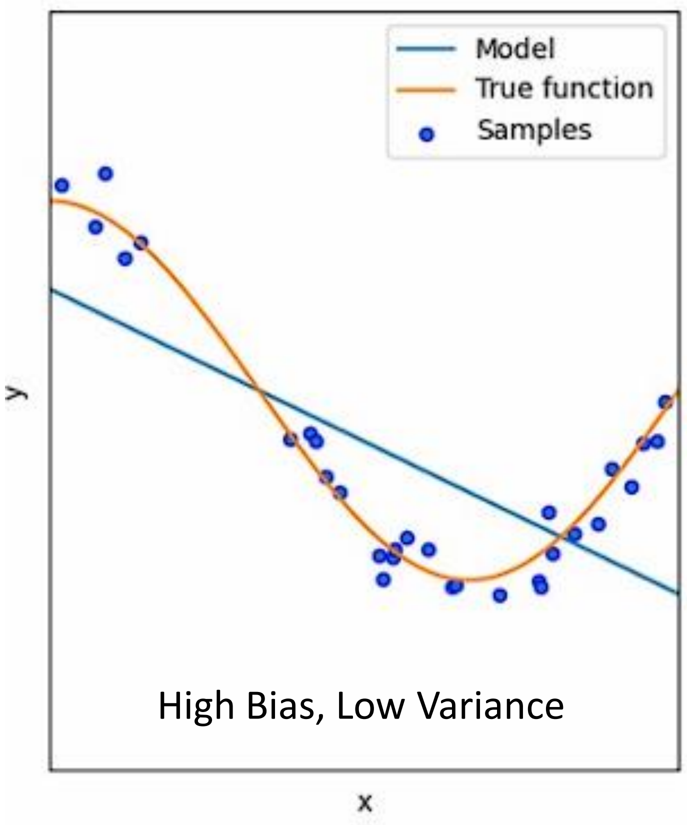


Fig 01: Underfitting

Training Performance: Good
Testing Performance: Good

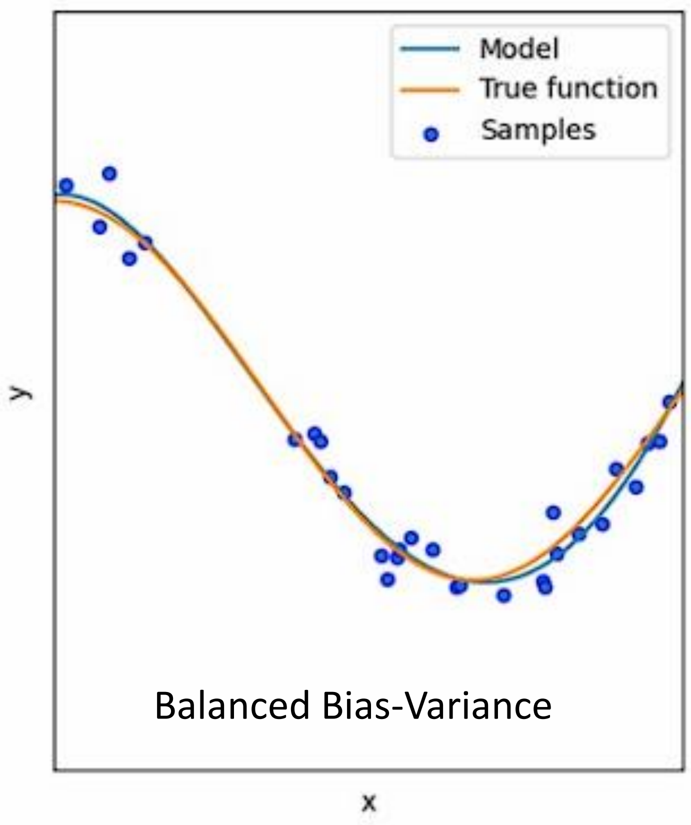


Fig 02: Best fitting

Training Performance: High (Rich)
Testing Performance: Low (Gorib)

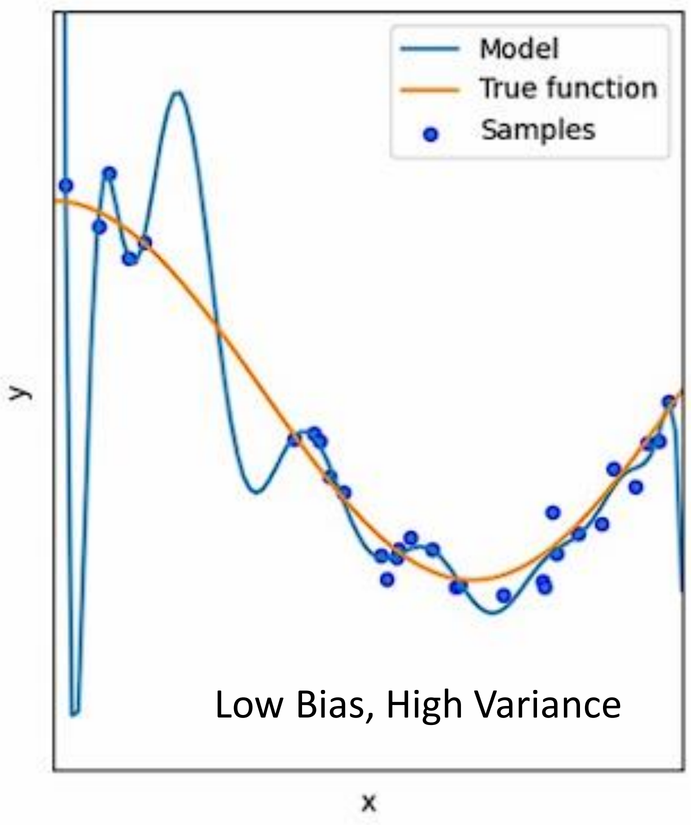


Fig 03: Overfitting

Overfitting and Underfitting

Why?

There are several reasons why a model might get overfitted, and it's helpful to understand the specifics of the situation to give you the most accurate answer. However, I can share some general causes and solutions:

- **Limited training data:** When you don't have enough data, the model can't learn the underlying patterns well enough to generalize to unseen data.
- **Model complexity:** Models with too many parameters or layers can easily overfit, memorizing specific details in the training data instead of learning generalizable rules.
- **Training for too long:** Overtraining makes the model focus on irrelevant details and noise in the training data, which leads to poor performance on new data.
- **Noisy data:** Data with errors or inconsistencies can mislead the model and cause overfitting.
- **Inappropriate features:** Using irrelevant or redundant features can increase the model's complexity and contribute to overfitting.

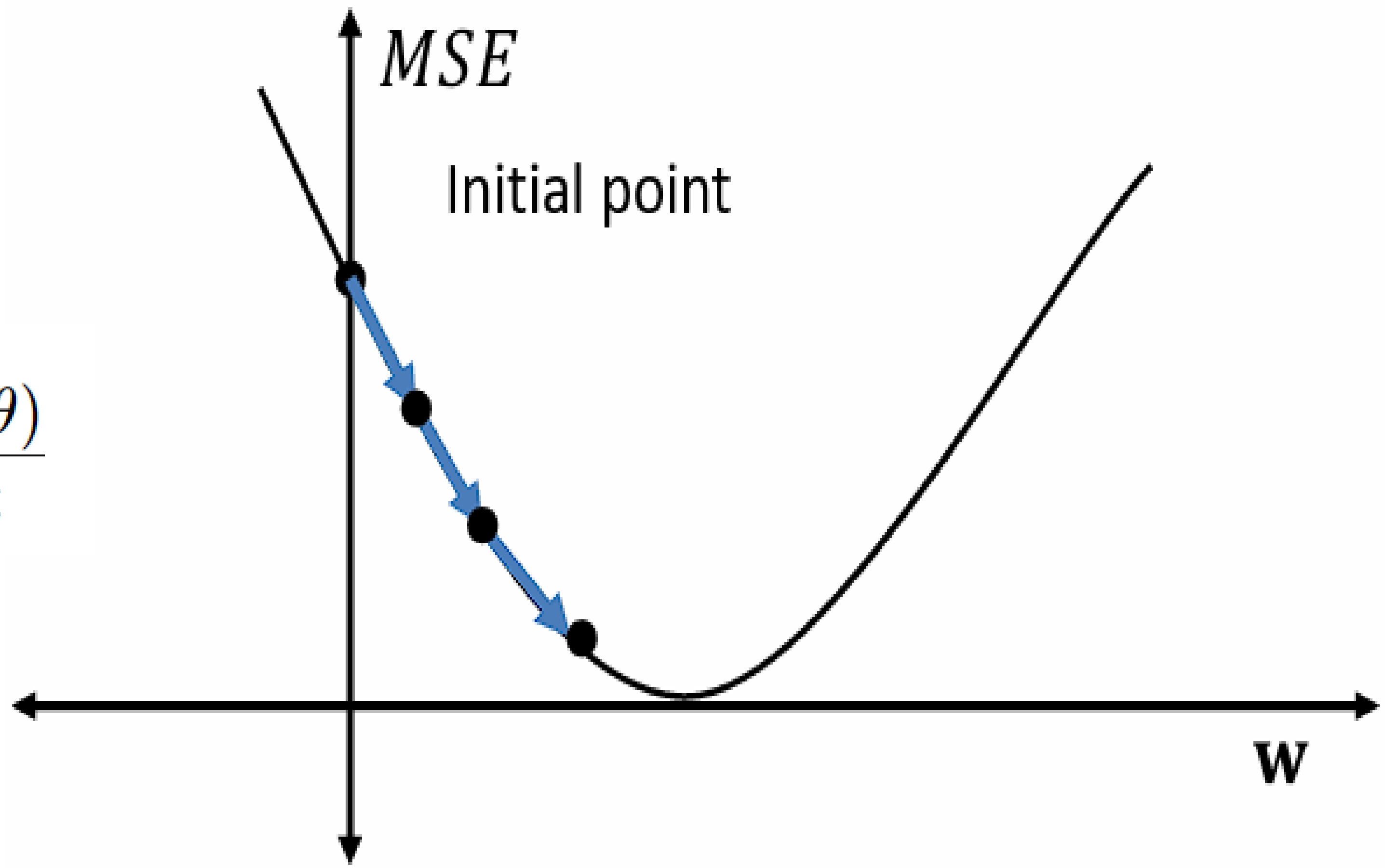
Overfitting and Underfitting

Solution?

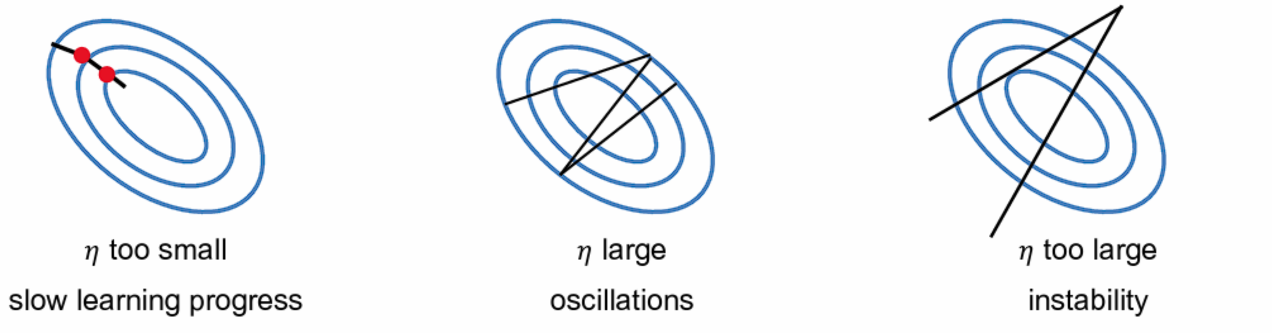
- **Increase the training data size:** Collect more data or use data augmentation techniques to create more samples from your existing data.
- **Simplify the model:** Reduce the number of parameters, layers, or features in your model.
- **Use regularization techniques:** L1 and L2 regularization penalize complex models, forcing them to learn simpler patterns.
- **Early stopping:** Stop training the model before it starts to overfit by monitoring its performance on a validation set.
- **Data cleaning and preprocessing:** Correct errors and inconsistencies in your data to avoid misleading the model.
- **Feature selection:** Choose only the most relevant features for your model to reduce complexity.

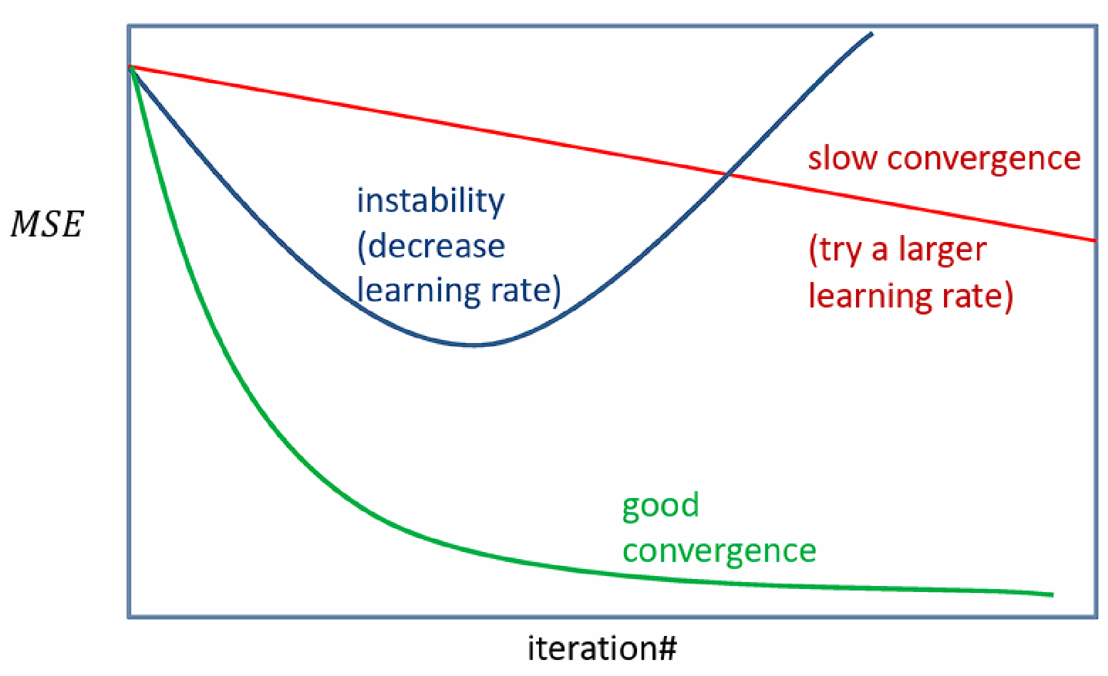
Update Parameter (weights):

$$\theta_i = \theta_{\text{old}} - \eta \frac{\partial J(\theta)}{\partial \theta_i}$$



The speed of the local/global minimum is affected from the learning rate.





The training curve could be used to monitor the effect of the learning rate value and observe the convergence. We plot the training loss against the number of training iterations.

The update rule for each parameter θ_i (where i indexes the parameters) in gradient descent can be represented as:

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \quad / \quad \theta_i = \theta_{\text{old}} - \eta \frac{\partial J(\theta)}{\partial \theta_i}$$

Where:

- α is the learning rate, a hyperparameter that controls the size of the steps taken during optimization.
- $J(\theta)$ is the cost function.
- $\frac{\partial J(\theta)}{\partial \theta_i}$ is the partial derivative of the cost function with respect to θ_i, which gives the gradient of the cost function with respect to that parameter.

Gradient descent continues to update the parameters until convergence, where the algorithm finds parameter values that minimize the cost function.

The Mean Squared Error (MSE) is calculated as the squared difference between the actual values ($y^{(i)}$) and the predicted values ($h_\theta(x^{(i)})$):

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

1. **Mean Squared Error (MSE):**
   - MSE quantifies the average of the squared differences between predicted values and actual values across the dataset.
   - The formula is expressed as: $\text{MSE} = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$, where:
     - y_i represents the actual value from the dataset.
     - $\hat{y}_i$ denotes the predicted value by the model.
     - n signifies the total number of data points.

2. **Substitution of $\hat{y}_i$ with $mx_i + c$**:
   - In linear regression, predictions ($\hat{y}_i$) can be computed using the equation of a straight line, $y = mx + c$.
   - Here, m denotes the slope of the line, x_i is the input feature value, and c represents the y-intercept.
3. **Revised MSE Formula**:
   - After replacing $\hat{y}_i$ with the linear equation $mx_i + c$, the MSE is recalculated as:
     - $\text{MSE} = \frac{1}{n} \sum_{i=0}^{n} (y_i - (mx_i + c))^2$
   - This adjusted formulation is utilized to evaluate the performance of a linear regression model, aiming to minimize the MSE by identifying optimal values for m and c through model training.

Step: 01

Gradient (m) = 0
Intercept (c) = 0
Learning Rate (L) = ~0.001

Step: 02

Calculate the partial derivative of the Cost function with respect to m. Let the partial derivative of the Cost function with respect to m be Dm.

$$D_m = \frac{\partial(Cost\ Function)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - (mx_i + c))$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - y_{i\,pred})$$

Step: 03

Similarly, let's find the partial derivative with respect to c. Let the partial derivative of the Cost function with respect to c be Dc.

$$D_c = \frac{\partial(Cost\ Function)}{\partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i - (mx_i + c))^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))$$

$$\frac{-2}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})$$

Calculus

Derivative	Integral (Antiderivative)
$\frac{d}{dx}n = 0$	$\int 0 \; dx = C$
$\frac{d}{dx}x = 1$	$\int 1 \, dx = x + C$
$\frac{d}{dx}x^n = nx^{n-1}$	$\int x^n \; dx = \frac{x^{n+1}}{n+1} + C$
$\frac{d}{dx}e^x = e^x$	$\int e^x \; dx = e^x + C$
$\frac{d}{dx}\ln x = \frac{1}{x}$	$\int \frac{1}{x} \; dx = \ln x + C$
$\frac{d}{dx}n^x = n^x \ln x$	$\int n^x \; dx = \frac{n^x}{\ln n} + C$
$\frac{d}{dx}\sin x = \cos x$	$\int \cos x \; dx = \sin x + C$
$\frac{d}{dx}\cos x = -\sin x$	$\int \sin x \; dx = -\cos x + C$

$\frac{d}{dx}\tan x = \sec^2 x$	$\int \sec^2 x \; dx = \tan x + C$
$\frac{d}{dx}\cot x = -\csc^2 x$	$\int \csc^2 x \; dx = -\cot x + C$
$\frac{d}{dx}\sec x = \sec x \tan x$	$\int \tan x \sec x \; dx = \sec x + C$
$\frac{d}{dx}\csc x = -\csc x \cot x$	$\int \cot x \csc x \; dx = -\csc x + C$
$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}}$	$\int \frac{1}{\sqrt{1-x^2}} \; dx = \arcsin x + C$
$\frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$	$\int -\frac{1}{\sqrt{1-x^2}} \; dx = \arccos x + C$
$\frac{d}{dx}\arctan x = \frac{1}{1+x^2}$	$\int \frac{1}{1+x^2} \; dx = \arctan x + C$
$\frac{d}{dx}\text{arc}\cot x = -\frac{1}{1+x^2}$	$\int -\frac{1}{1+x^2} \; dx = \text{arc}\cot x + C$
$\frac{d}{dx}\text{arc}\sec x = \frac{1}{x\sqrt{x^2-1}}$	$\int \frac{1}{x\sqrt{x^2-1}} \; dx = \text{arc}\sec x + C$
$\frac{d}{dx}\text{arc}\csc x = -\frac{1}{x\sqrt{x^2-1}}$	$\int -\frac{1}{x\sqrt{x^2-1}} \; dx = \text{arc}\csc x + C$

Step: 04

Update the value of the gradient and intercept.

$$\text{new}_m = \text{old}_m - \text{Learning rate} \times D_m$$
$$\text{new}_c = \text{old}_c - \text{Learning rate} \times D_c$$

Repeat the steps!
500-1000 times

Linear Regression with Multiple Variables

Linear Regression with Multiple Variables

Mathematical Representation

predictor, 'x-variable', independent variable, explanatory variable

coefficient

$$Y = \underbrace{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p}_{\text{linear predictor}}$$

response, dependent variable, observation, 'y-variable'

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = expanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

Linear Regression with Single Vs. Multiple Variables

Mathematical Representation

Single $y = b_0 + b_1{*}x_1$

Dependent variable (DV) Independent variables (IVs)

Multiple $y = b_0 + b_1{*}x_1 + b_2{*}x_2 + \ldots + b_n{*}x_n$

R Squared Value / Model Accuracy

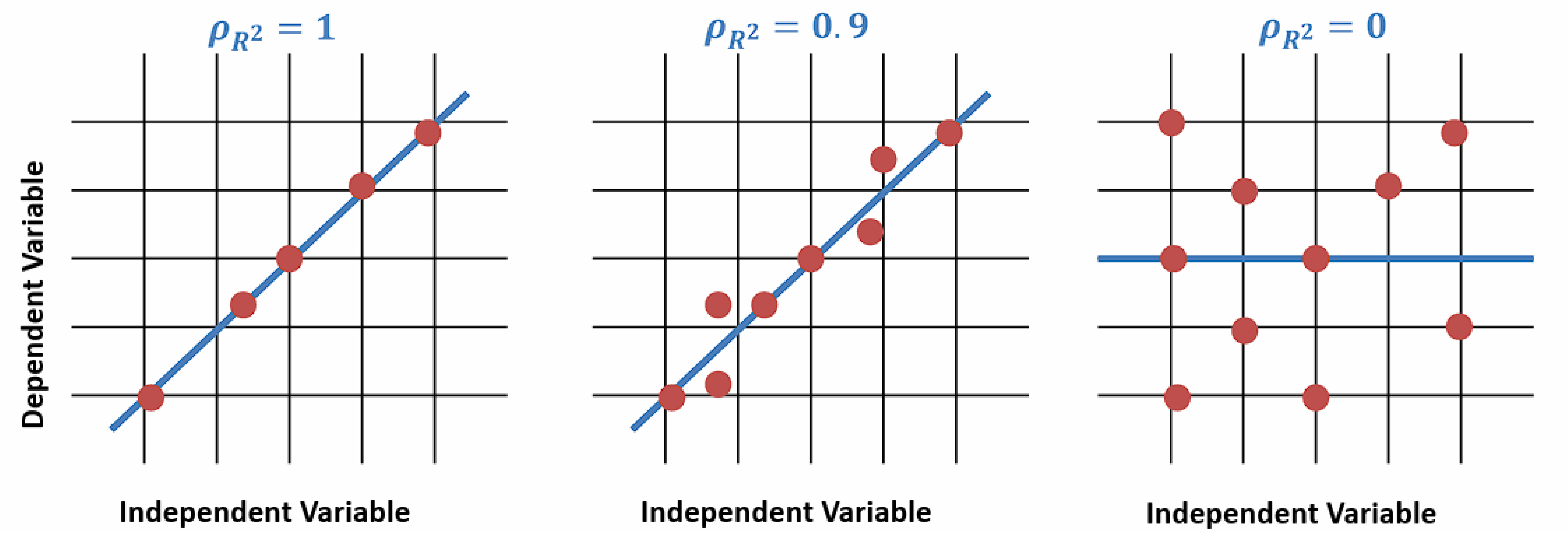
Mathematical Calculation

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

- Residual sum of squared errors of our regression model (SSres)
- Total sum of squared errors (SStot)

R Squared Value / Model Accuracy

Mathematical Calculation



R Squared Value / Model Accuracy

Python Implementation

Way no: 01

```
reg.score(xtest, ytest)
```

Way no: 02

```
y_pred = reg.predict(xtest) #Predicted y
from sklearn.metrics import r2_score
Score = r2_score(ytest, y_pred)
```