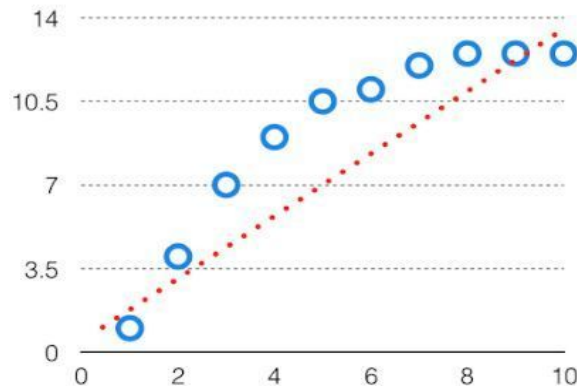


# Bias-Variance Tradeoff – Machine Learning

Understanding bias and variance is crucial for accuracy in machine learning. The **bias-variance tradeoff** helps balance the model's ability to minimize both errors. A proper understanding of these errors helps avoid **overfitting** (high variance) and **underfitting** (high bias) while training the model, ensuring optimal performance.

## What is Bias?

The bias is known as the difference between the prediction of the values by the [Machine Learning](#) model and the correct value. Being high in biasing gives a large error in training as well as testing data. It is recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the [Underfitting of Data](#). This happens when the [hypothesis](#) is too simple or linear in nature. Refer to the graph given below for an example of such a situation.



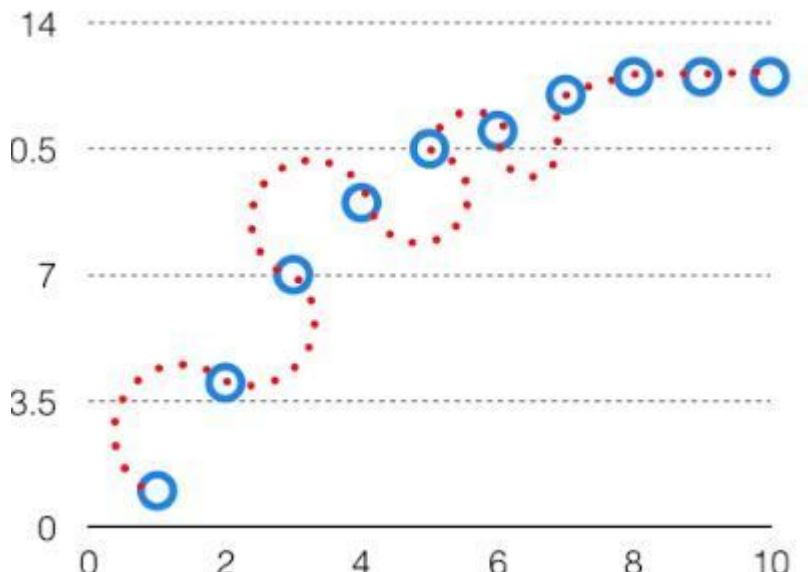
*High Bias in the Model*

In such a problem, a hypothesis looks like follows.

$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

## What is Variance?

The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but have high error rates on test data. When a model is high on variance, it is then said to as **Overfitting of Data**. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low. The high variance data looks as follows.



*High Variance in the Model*

In such a problem, a hypothesis looks like follows.

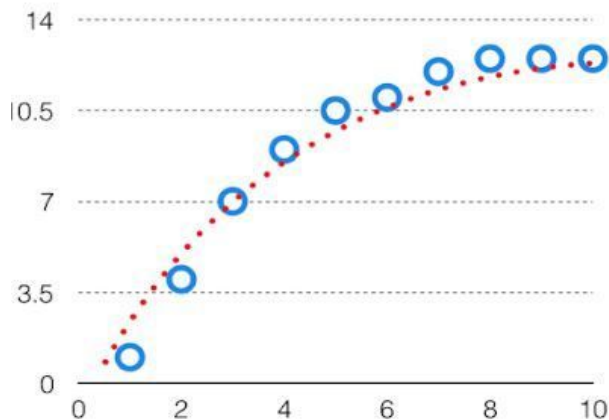
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4)$$

# Bias Variance Tradeoff

The **Bias-Variance Tradeoff** explains the balance between a model's simplicity and complexity:

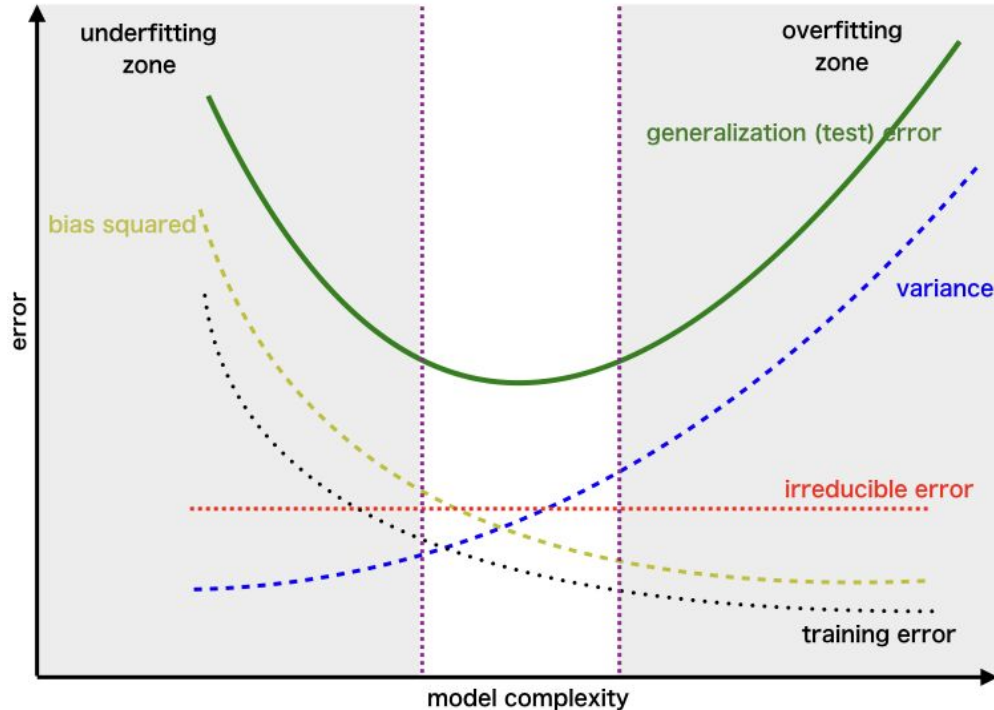
- **High bias, low variance:** If the model is too simple (e.g., a linear equation), it may underfit the data, leading to poor performance on both training and testing data.
- **Low bias, high variance:** If the model is too complex (e.g., a high-degree polynomial), it may overfit the data, performing well on training data but poorly on new, unseen data.

The tradeoff means that a model cannot be both highly complex and simple at the same time. The ideal solution lies between these extremes, where bias and variance are balanced to achieve the best generalization performance.



We try to optimize the value of the total error for the model by using the [Bias-Variance](#) Tradeoff.

The best fit will be given by the hypothesis on the tradeoff point. The error to complexity graph to show trade-off is given as –



This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

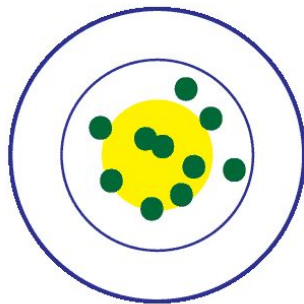
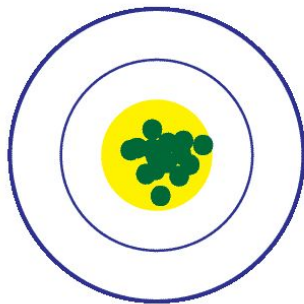
## Different Combinations of Bias-Variance

- **High Bias, Low Variance:** A model that has high bias and low variance is considered to be underfitting.
- **High Variance, Low Bias:** A model that has high variance and low bias is considered to be overfitting.
- **High-Bias, High-Variance:** A model with high bias and high variance cannot capture underlying patterns and is too sensitive to training data changes. On average, the model will generate unreliable and inconsistent predictions.
- **Low Bias, Low Variance:** A model with low bias and low variance can capture data patterns and handle variations in training data. This is the perfect scenario for a machine learning model where it can generalize well to unseen data and make consistent, accurate predictions. However, in reality, this is not feasible.

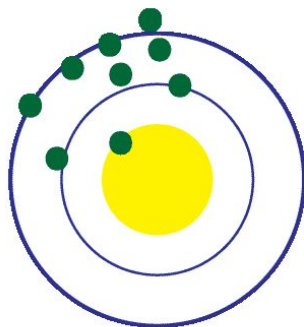
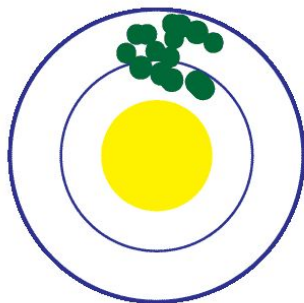
**Low Variance**

**High Variance**

**Low Bias**



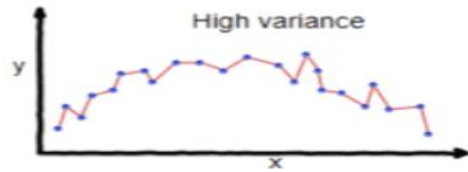
**High Bias**



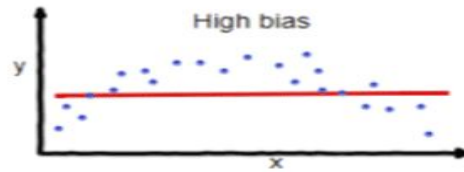
# Regularization in Machine Learning

Regularization is a technique used to reduce errors by fitting the function appropriately on the given training set and avoiding overfitting. The commonly used [regularization techniques](#) are :

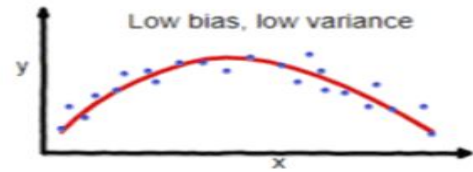
1. Lasso Regularization – L1 Regularization
2. Ridge Regularization – L2 Regularization
3. Elastic Net Regularization – L1 and L2 Regularization



**overfitting**



**underfitting**



**Good balance**



## 1. Lasso Regression (L1 Regularization)

Lasso (Least Absolute Shrinkage and Selection Operator) adds a penalty to the loss function based on the **absolute values** of the model's coefficients. This encourages sparsity, meaning

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^m |w_i|$$

where,

- $m$  – Number of Features
- $n$  – Number of Examples
- $y_i$  – Actual Target Value
- $y_i(\text{hat})$  – Predicted Target Value

**Example:** If you have a dataset with 100 features, Lasso regression will reduce the coefficients of less important features to zero, keeping only the relevant features in the model. This leads to a simpler and more interpretable model.

## 2. Ridge Regression (L2 Regularization)

Ridge regression adds a penalty based on the **squared values** of the model's coefficients. Unlike Lasso, Ridge doesn't perform feature selection, but it reduces the size of the coefficients, preventing the model from becoming too complex.

**Ridge Cost Function:**

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^m w_i^2$$

**Example:** Ridge regression might be useful when you have many features that are all important, but some may have large weights that lead to overfitting. By adding a penalty for large weights, Ridge helps to ensure that the model doesn't overly rely on any one feature.

### 3. Elastic Net Regression

Elastic Net is a combination of both **L1 and L2 regularization**. It uses a mix of absolute and squared magnitudes of the coefficients, controlled by an additional hyperparameter  $\alpha$ , which adjusts the balance between Lasso and Ridge.

**Elastic Net Cost Function:**

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \left( (1 - \alpha) \sum_{i=1}^m |w_i| + \alpha \sum_{i=1}^m w_i^2 \right)$$

Where:

- $\alpha$  controls the mixing ratio between Lasso and Ridge.

**Example:** If you are unsure whether Lasso or Ridge works better for your dataset, you can use Elastic Net to combine the benefits of both. The model can select features like Lasso, but also shrink the coefficients like Ridge.


## Benefits of Regularization

- **Prevents overfitting:** Regularization helps the model generalize better by avoiding complex, overfitted models.
- **Improves interpretability:** Lasso, in particular, reduces the number of features, making the model simpler and easier to interpret.
- **Handles multicollinearity:** Regularization helps reduce issues with highly correlated features.
- **Stabilizes models:** By penalizing large coefficients, regularization makes the model more stable, especially when there is limited or noisy data.
- **Fine-tuning model performance:** Regularization introduces a hyperparameter (e.g.,  $\lambda$  or  $\alpha$ ) to control the strength of regularization, allowing you to adjust the model to achieve the best balance between bias and variance.

## Example: Regularization in Action

Consider a dataset with a high-dimensional feature set, where some features are irrelevant or noisy. Without regularization, a linear regression model might overfit the data by giving large weights to the irrelevant features. After applying **Lasso regularization**, the coefficients of those irrelevant features are driven to zero, effectively removing them from the model. This leads to a simpler model that performs better on unseen data, as it has not "memorized" the noise from the training data.

In summary, regularization techniques like Lasso, Ridge, and Elastic Net improve a model's generalization by penalizing large or irrelevant coefficients, preventing overfitting, and enhancing stability across different datasets.

Aspect	L1 Regularization (Lasso)	L2 Regularization (Ridge)
Penalty	Adds the absolute value of coefficients to the cost function.	Adds the squared value of coefficients to the cost function.
Mathematical Term	$(\lambda \sum_{i=1}^m  w_i )$	$\sum w_i^2$
Effect on Coefficients	Drives some coefficients to exactly zero, leading to feature selection.	Shrinks coefficients but rarely reduces them to zero.
Sparsity	Produces sparse models by excluding irrelevant features.	Produces non-sparse models where all features contribute.
Use Case	Useful when you need feature selection or interpretability.	Useful when you need to retain all features and reduce overfitting.
Optimization	Solved using methods like coordinate descent.	Solved using gradient-based optimization.
Shape of Penalty	Diamond-shaped constraint region (sharp corners).	Circular constraint region (smooth edges).
Handling Multicollinearity	Can randomly select one feature when features are highly correlated. 	Distributes weights among correlated features.