

Welcome...

- Regression: Linear Regression, Ridge & Lasso Regression

CS 797Q
Fall 2024

10/28/2024



Categories of Data Analysis Techniques

Category	Techniques Covered	Problem to be solved
Association Rules	Apriori	Relationships between items
Clustering	K-Means Clustering DB Scan	Grouping of similar items Identification of structures
Classification	K-nearest Neighbor Decision Trees Random Forests Logistic Regression Naive Bayes Support Vector Machines Neural Networks	Assignment of labels to objects
Regression	Linear Regression Ridge Lasso	Relationship between outcome and inputs
Time Series Analysis	ARMA	Identification of temporal structures Forecasting of temporal processes
Text Mining	Bag-of-Words Stemming/Lemmatization TF-IDF	Analysis of textual data

Linear Regression



Linear regression is a supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.



When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.



Similarly, when there is only one dependent variable, known as Univariate Linear Regression, while when there are more than one dependent variables, it is known as Multivariate Regression.

Types of Linear Regression

➤ Simple Linear Regression

it involves only one independent variable and one dependent variable. The equation for simple linear regression is:

$$Y = \beta_0 + \beta_1 X$$

- Where Y is the dependent variable
- X is the independent variable
- β_0 is the intercept
- β_1 is the slope

➤ Multiple Linear Regression

This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

where:

- Y is the dependent variable
- X_1, X_2, \dots, X_n are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \dots, \beta_n$ are the slopes

Hypothesis function in Linear Regression

- Assumed that independent feature is experience i.e., X and respective salary Y is the dependent variable. Let's assume there is a linear relationship between X and Y then the salary can be predicted using

$$\hat{Y} = \theta_1 + \theta_2 X$$

OR

$$\hat{y}_i = \theta_1 + \theta_2 x_i$$

Here,

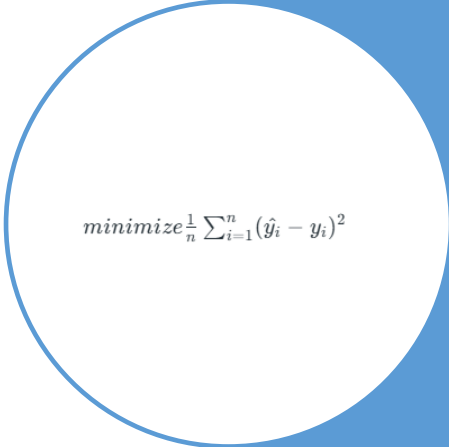
- $y_i \in Y$ ($i = 1, 2, \dots, n$) are labels to data (Supervised learning)
- $x_i \in X$ ($i = 1, 2, \dots, n$) are the input independent training data (univariate – one input variable(parameter))
- $\hat{y}_i \in \hat{Y}$ ($i = 1, 2, \dots, n$) are the predicted values.

The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

- θ_1 : intercept
- θ_2 : coefficient of x

How to update θ_1 and θ_2 values to get the best-fit line?

- To achieve the best-fit regression line, the model aims to predict the target value \hat{Y} such that the error difference between the predicted value \hat{Y} and the true value Y is minimum.
- So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimizes the error between the predicted value (pred) and the true value.


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

Cost function for Linear Regression



The cost function or the loss function is nothing but the error or difference between the predicted value \hat{Y} and the true value Y .



In Linear Regression, the **Mean Squared Error (MSE)** cost function is employed, which calculates the average of the squared errors between the predicted values \hat{y}_i and the actual values Y_i .



The purpose is to determine the optimal values for the intercept θ_1 and the coefficient of the input feature θ_2 ; The linear equation expressing this relationship is $\hat{y}_i = \theta_1 + \theta_2 x_i$

Cost function for Linear Regression



MSE function can be calculated as:



Cost function(J)= $\frac{1}{n} \sum (y_i^{\wedge} - y_i)^2$



Utilizing the MSE function, iterative process of gradient descent is applied to update the values of θ_1 & θ_2 . This ensures that the MSE value converges to the global minima, signifying the most accurate fit of the linear regression line to the dataset.



This process involves continuously adjusting the parameters θ_1 & θ_2 based on the gradients calculated from the MSE.

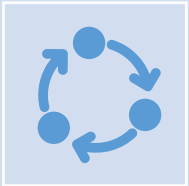


The result is a linear regression line that minimizes the overall squared differences between the predicted and actual values, providing an optimal representation of the underlying relationship in the data.

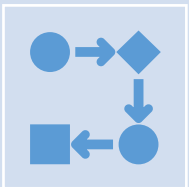
Gradient Descent Algorithm



A linear regression model can be trained using the optimization algorithm gradient descent by iteratively modifying the model's parameters to reduce the mean squared error (MSE) of the model on a training dataset.



To update θ_1 and θ_2 values to reduce the Cost function (minimizing RMSE value) and achieve the best-fit line the model uses Gradient Descent.



The idea is to start with random θ_1 and θ_2 values and then iteratively update the values, reaching minimum cost.

Gradient Descent Algorithm

Let's differentiate the cost function(J) with respect to θ_1

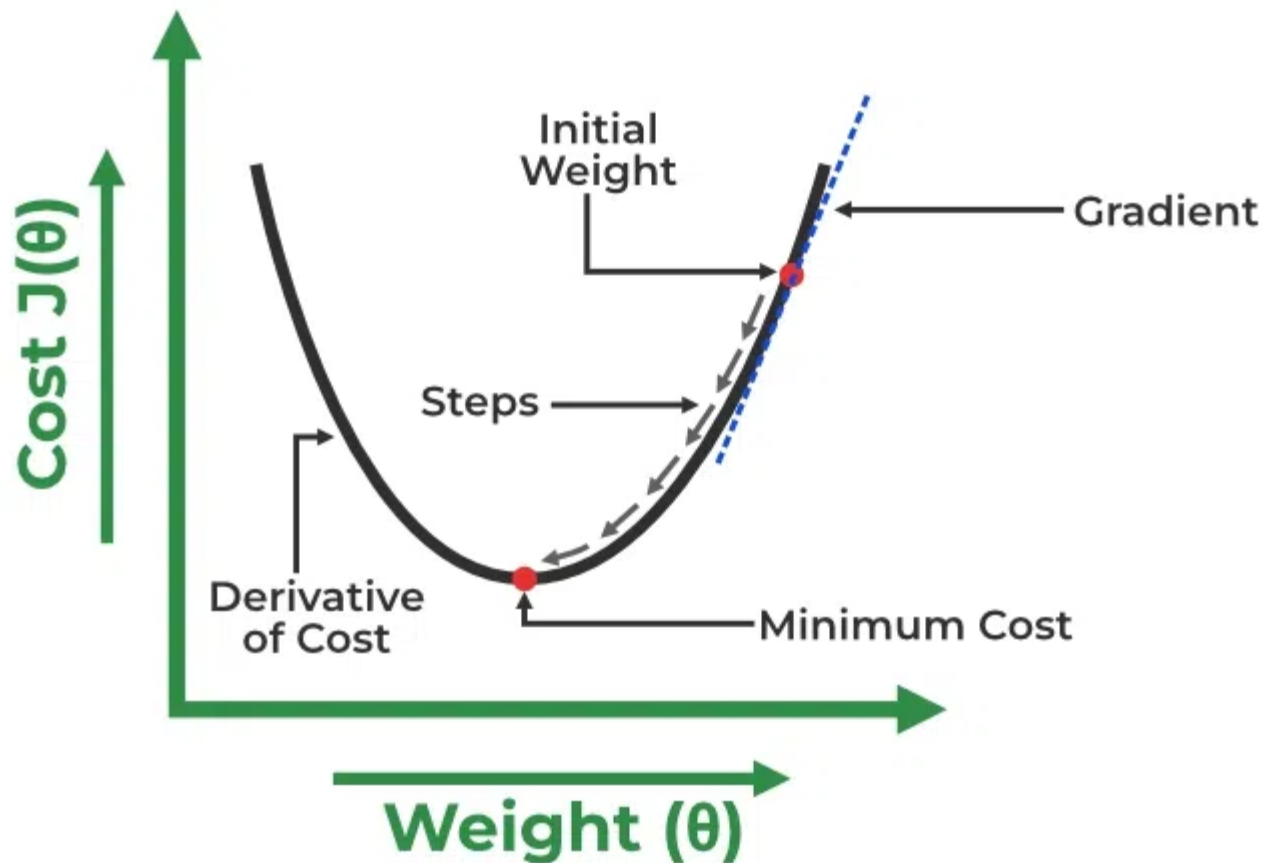
$$\begin{aligned} J'_{\theta_1} &= \frac{\partial J(\theta_1, \theta_2)}{\partial \theta_1} \\ &= \frac{\partial}{\partial \theta_1} \left[\frac{1}{n} \left(\sum_{i=1}^n (\hat{y}_i - y_i)^2 \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_1} (\hat{y}_i - y_i) \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_1} (\theta_1 + \theta_2 x_i - y_i) \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) (1 + 0 - 0) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n (\hat{y}_i - y_i) (2) \right] \\ &= \frac{2}{n} \sum_{i=1}^n (\hat{y}_i - y_i) \end{aligned}$$

$$\begin{aligned} \theta_1 &= \theta_1 - \alpha (J'_{\theta_1}) \\ &= \theta_1 - \alpha \left(\frac{2}{n} \sum_{i=1}^n (\hat{y}_i - y_i) \right) \\ \theta_2 &= \theta_2 - \alpha (J'_{\theta_2}) \\ &= \theta_2 - \alpha \left(\frac{2}{n} \sum_{i=1}^n (\hat{y}_i - y_i) \cdot x_i \right) \end{aligned}$$

Let's differentiate the cost function(J) with respect to θ_2

$$\begin{aligned} J'_{\theta_2} &= \frac{\partial J(\theta_1, \theta_2)}{\partial \theta_2} \\ &= \frac{\partial}{\partial \theta_2} \left[\frac{1}{n} \left(\sum_{i=1}^n (\hat{y}_i - y_i)^2 \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_2} (\hat{y}_i - y_i) \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) \left(\frac{\partial}{\partial \theta_2} (\theta_1 + \theta_2 x_i - y_i) \right) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n 2(\hat{y}_i - y_i) (0 + x_i - 0) \right] \\ &= \frac{1}{n} \left[\sum_{i=1}^n (\hat{y}_i - y_i) (2x_i) \right] \\ &= \frac{2}{n} \sum_{i=1}^n (\hat{y}_i - y_i) \cdot x_i \end{aligned}$$

Gradient Descent Algorithm



Evaluation Metrics for linear regression

Mean Square Error (MSE)

- Mean Squared Error (MSE) calculates the average of the squared differences between the actual and predicted values for all the data points.

$$MSE = 1/n \sum (y_i^{\wedge} - y_i)^2$$

- Here, n is the number of data points.
- y_i is the actual or observed value for the i^{th} data point.
- y_i^{\wedge} is the predicted value for the i^{th} data point.
- MSE is a way to quantify the accuracy of a model's predictions.

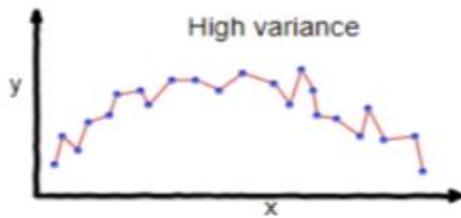
Mean Absolute Error

- Mean Absolute Error (MAE) measures the average absolute difference between the predicted values and actual values.
- MAE is expressed as:
- Here, n is the number of observations
- Y_i represents the actual values.
- \hat{Y}_i represents the predicted values
- Lower MAE value indicates better model performance.

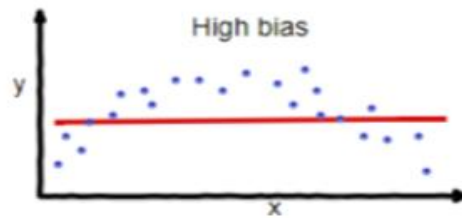
$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$$

Lasso vs Ridge

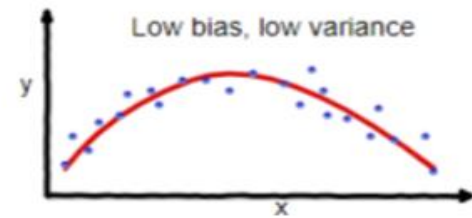
- **Bias:** Bias is the model's tendency to make assumptions that simplify learning. While it can help generalize the data, high bias may lead to underfitting, where the model is too simple and misses important patterns.
- **Variance:** **Variance** is the error from a model being too sensitive to small fluctuations in the data, often causing it to capture noise or outliers (overfitting). High-variance models perform well on training data but poorly on new data. Examples include Decision Trees and KNN.



overfitting



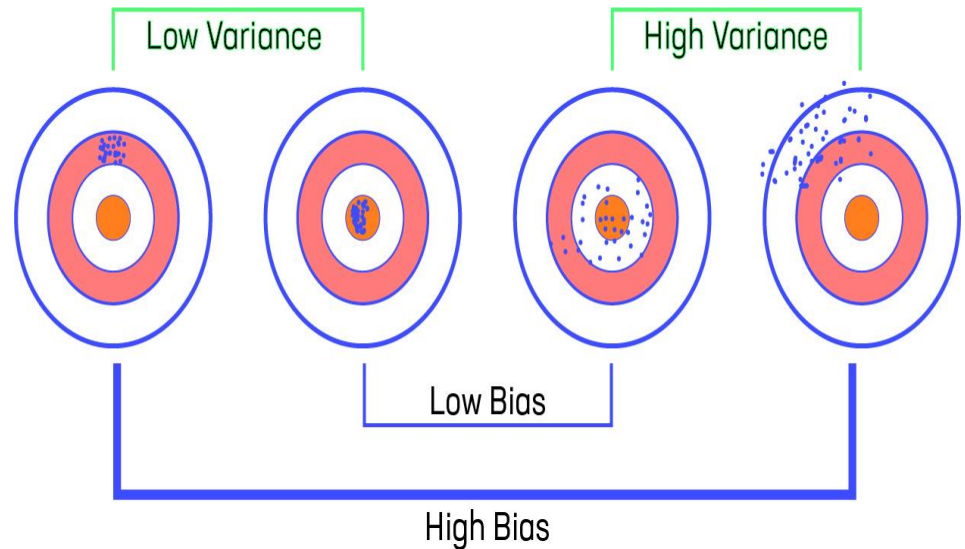
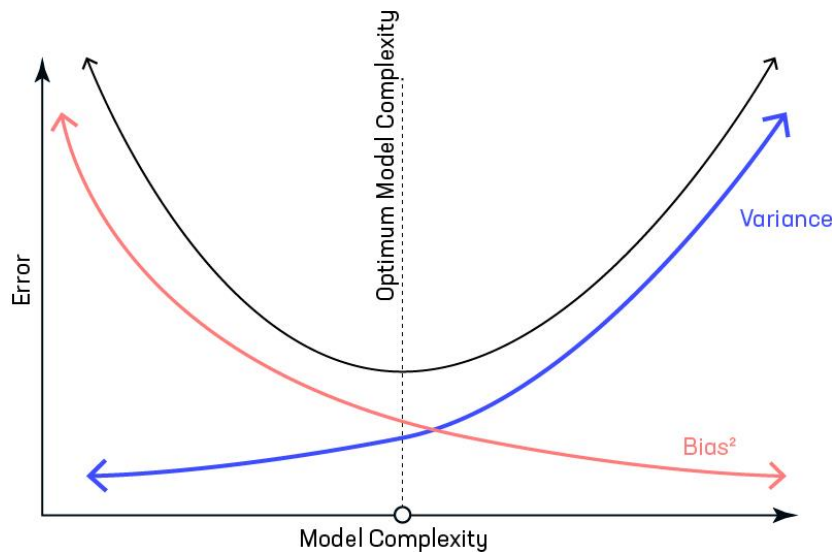
underfitting



Good balance

Variance and Bias

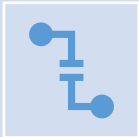
- Let us consider that we have a very accurate model, this model has a low error in predictions. This model has low bias and variance.
- Now, if the predictions are scattered here and there then that is the symbol of high variance, also if the predictions are far from the target, then that is the symbol of high bias.
- Sometimes we need to choose between low variance and low bias. There is an approach that prefers some bias over high variance, this approach is called **Regularization**.



Ridge Regression



In Ridge regression, the L2 penalty is added to the model's cost function. This penalty is the sum of the squares of the model's coefficient values.



Purpose: The penalty discourages the model from having large coefficient values, which helps it generalize better to new data. Large coefficients can make the model too complex and more likely to overfit (perform well on training data but poorly on new data).



Effect of Lambda (λ): The parameter λ (lambda) controls the strength of this penalty. When λ is high, the penalty is stronger, pushing the coefficients closer to zero. When λ is low, the penalty is weaker, allowing the coefficients to take larger values.



Result: By limiting the size of the coefficients, Ridge regression creates a simpler model that generalizes better, reducing the risk of overfitting. In Ridge regression, the L2 penalty is added to the model's cost function. This penalty is the sum of the squares of the model's coefficient values.

Lasso Regression

In Lasso regression, the L1 penalty is added to the cost function. This penalty is the sum of the absolute values of the model's coefficients.

Purpose: Like Ridge regression, Lasso discourages large coefficient values, helping the model avoid overfitting.

But Lasso has an additional benefit: it can reduce some coefficients exactly to zero, effectively selecting only the most important features.

Effect of Lambda (λ): The parameter λ (lambda) controls how strong the penalty is. When λ is high, the penalty is stronger, which pushes more coefficients to zero. When λ is low, the penalty is weaker, allowing more coefficients to remain non-zero.