Machine Learning
ICT-4261

By-
Dr. Jesmin Akhter
Professor
Institute of Information Technology
Jahangirnagar University

Contents

The course will mainly cover the following topics:

- ✓ A Gentle Introduction to Machine Learning
- ✓ Linear Regression
- ✓ Logistic Regression
- ✓ Naive Bayes
- ✓ Support Vector Machines
- ✓ Decision Trees and Ensemble Learning
- ✓ Clustering Fundamentals
- ✓ Hierarchical Clustering
- ✓ Neural Networks and Deep Learning
- ✓ Unsupervised Learning

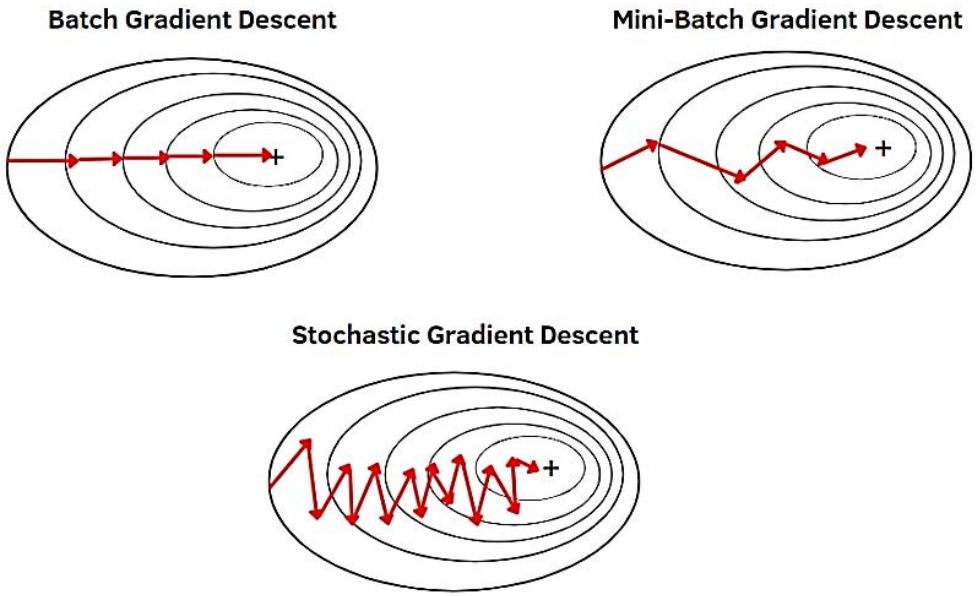
Outline

- ✓ Linear Regression
  - Different Types of Gradient Descent Algorithms
  - Ridge, Lasso

Different Types of Gradient Descent Algorithms

Gradient descent algorithm can be divided into three categories

1. **Batch Gradient Descent**
2. **Stochastic Gradient Descent (SGD)**
3. **Mini Batch Gradient Descent-**In



Batch Gradient Descent

- ✓ In batch gradient descent, the update of model parameters or weights is calculated using the entire training dataset at once. This means that for each step in the training process, the algorithm calculates the gradient of the cost function for the entire dataset.
- ✓ This algorithm is often used when the **training dataset is relatively small** and can fit into memory comfortably.
- ✓ For example, consider a linear regression model where you're predicting housing prices based on features like size and location. If your **dataset is small**, you could use batch gradient descent to train your model, updating the weights based on the error calculated from the entire dataset.

Repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

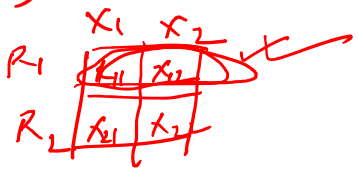
Batch Gradient Descent

- ✓ Despite its accuracy and consistency, batch gradient descent has several drawbacks.
  - The batch gradient descent method typically requires the entire training dataset in memory and is implemented for use in the algorithm.
  - Large datasets can result in very slow model updates or training speeds.
  - It can be slow and computationally expensive(require more computational power) when dealing with large datasets and complex models.
  - Additionally, it can get stuck in shallow local minima or saddle points, where the gradient is either very small or zero.
  - Furthermore, it does not permit online or incremental learning, meaning new data cannot be added to the training set without restarting the algorithm.

Loss function for multiple linear regression –

MSE, $L = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$

$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n - y_i)^2$ For n features



Now for simplicity for 2 features

$$L = \frac{1}{2} \cdot \frac{1}{2} \left[(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)^2 + (\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2 \right]$$

Let ~~$h_\theta(x_i) = \hat{y}_1 = \theta_0 + \theta_1$~~

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

$\hat{y}_1 = \theta_0 + \theta_1 x_{11} + \theta_2 x_{12}$

$\hat{y}_2 = \theta_0 + \theta_1 x_{21} + \theta_2 x_{22}$

Again, $L = \frac{1}{2} \cdot \frac{1}{2} \left[(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)^2 + (\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2 \right]$

$$\frac{\delta L}{\delta \theta_0} = \frac{1}{2} \cdot \frac{1}{2} \left[2(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)(1) + 2(\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)(1) \right]$$

$$= \frac{1}{2} \cdot \frac{1}{2} \left[2(\hat{y}_1 - y_1) + 2(\hat{y}_2 - y_2) \right]$$

$$= \frac{1}{2} \cdot \frac{2}{2} \left[(\hat{y}_1 - y_1) + (\hat{y}_2 - y_2) \right]$$

Now for m samples

$$\frac{\delta L}{\delta \theta_0} = \frac{1}{2} \cdot \frac{2}{m} \left[(\hat{y}_1 - y_1) + (\hat{y}_2 - y_2) + (\hat{y}_3 - y_3) + \cdots + (\hat{y}_m - y_m) \right]$$

$$\frac{\delta L}{\delta \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)$$

$$\theta_0 = \theta_0 - \alpha \frac{\delta L}{\delta \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\delta L}{\delta \theta_1}$$

$$\theta_2 = \theta_2 - \alpha \frac{\delta L}{\delta \theta_2}$$

$$\theta_j = \theta_j - \alpha \frac{\delta L}{\delta \theta_j} \quad \text{where} \quad j = 0, 1, 2, \cdots n$$

Again, $L = \frac{1}{2} \cdot \frac{1}{2} \sum_{i=1}^{m=2} (\hat{y}_i - y_i)^2$

$= \frac{1}{2} \cdot \frac{1}{2} [(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2]$

$= \frac{1}{2} \cdot \frac{1}{2} [(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)^2 + (\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2]$

$\frac{\partial L}{\partial \theta_1} = \frac{1}{2} \cdot \frac{1}{2} [(\hat{y}_1 - y_1)(x_{11}) + 2 (\hat{y}_2 - y_2)(x_{21})]$

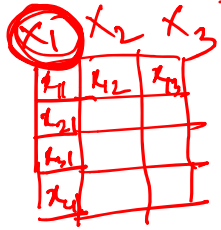
$\frac{\partial L}{\partial \theta_1} = \frac{1}{2} \cdot \frac{2}{m} [(\hat{y}_1 - y_1)(x_{11}) + (\hat{y}_2 - y_2)(x_{21}) + (\hat{y}_3 - y_3)(x_{31}) + \cdots + (\hat{y}_n - y_n)\, x_{m1}]$ for m feature records.

$L = \theta_1 x_1$

$\frac{\partial L}{\partial \theta_1} = x_1$

$\theta_1 x_1$

x_1



$= (x_{11} + x_{21} + x_{31} + \cdots)$

$$\frac{\delta L}{\delta \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{i1})$$

$$\frac{\delta L}{\delta \theta_2} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{i2})$$

$\therefore$

$$\vdots$$

$$\frac{\delta L}{\delta \theta_n} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)\, x_{in}$$

Now, $\theta_j = \theta_j - \alpha \frac{\delta L}{\delta \theta_j}$ where, $j = 0, 1, 2, \cdots n$

$$= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{ij}$$ where, $j = 0, 1, 2, \cdots n$

Finally Repeat until convergence

{

$$\theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{ij}$$

} where, $j = 0, 1, 2, \cdots n$

Repeat until convergence

{

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{ij}$$

}

where, $j = 0, 1, 2, \cdots n$

For,
$m = 2$
$n = 2$

Now update
θ_1, θ_2 -- -
steps 1.
Taking random
values $\theta_0 = 0$, $\theta_1 = 1$,
$\theta_2 = 1$
steps 2. epoch = 100
$\alpha = 0.1$ ...

Repeat until convergence

{

$$\theta_0 = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 = \theta_1 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i1}$$

$$\theta_2 = \theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i2}$$

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{in}$$

Stochastic Gradient Descent (SGD)

- ✓ Stochastic gradient descent updates the model's parameters using only one training example or a small batch of examples at a time. This means the gradient and hence the parameter update is calculated and applied after each example or small batch, making the process more incremental.
- ✓ SGD is particularly useful when dealing with large datasets that cannot fit into memory. For example, the model parameters are updated incrementally as each image (or a small batch of images) is processed, enabling the model to learn progressively without the need to load the entire dataset into memory.
- ✓ We update the values of parameters after each example of training set. It is relatively faster than other types but its accuracy is less.

$$for\ i\ in\ range\ (m):$$

$$\theta_j = \theta_j - \alpha\,(\hat{y}^i - y^i)\,x_j^i$$

Stochastic Gradient Descent (SGD)

Advantages

- ✓ You can instantly see your model's performance and improvement rates with frequent updates.
- ✓ This variant of the steepest descent method is probably the easiest to understand and implement, especially for beginners.
- ✓ Increasing the frequency of model updates will allow you to learn more about some issues faster.
- ✓ The noisy update process allows the model to avoid local minima (e.g., premature convergence).
- ✓ Faster and require less computational power.
- ✓ Suitable for the larger dataset.

Disadvantages

- ✓ Frequent model updates are more computationally intensive than other steepest descent configurations, and it takes considerable time to train the model with large datasets.
- ✓ Frequent updates can result in noisy gradient signals. This can result in model parameters and cause errors to fly around (more variance across the training epoch).
- ✓ A noisy learning process along the error gradient can also make it difficult for the algorithm to commit to the model's minimum error.

Mini Batch Gradient Descent-In

- ✓ In mini batch we divide the data set into certain number of batches of equal size so that we can update the parameters of the equation after the completion of each batch. This has moderate accuracy and time consumption.

Say $b = 10, m = 1000.$

Repeat {

for $i = 1, 11, 21, 31, \ldots, 991$ {

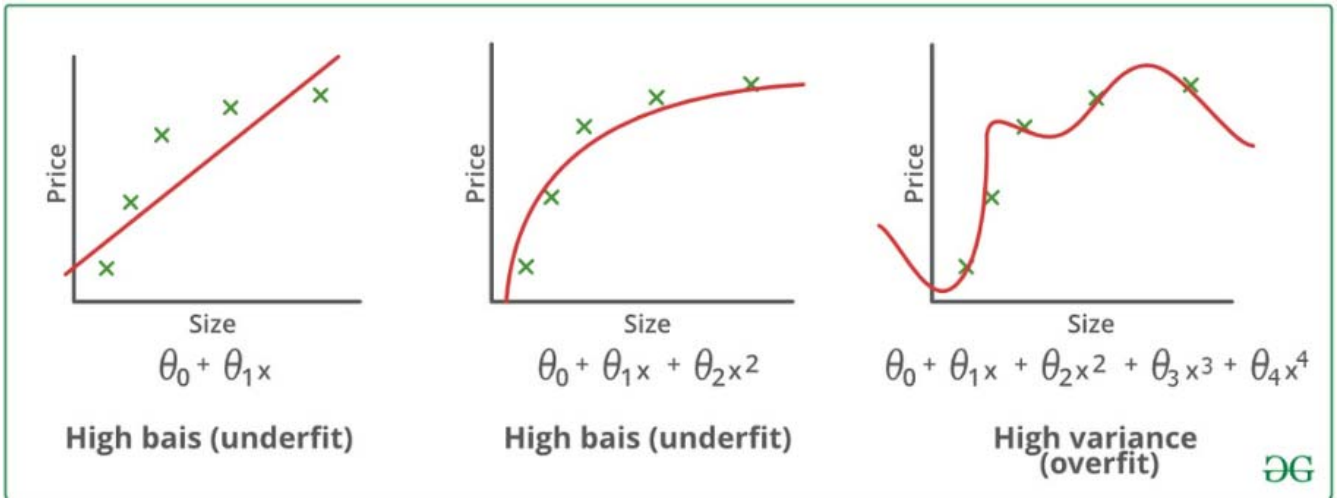
$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for every $j = 0, \ldots, n$) } }

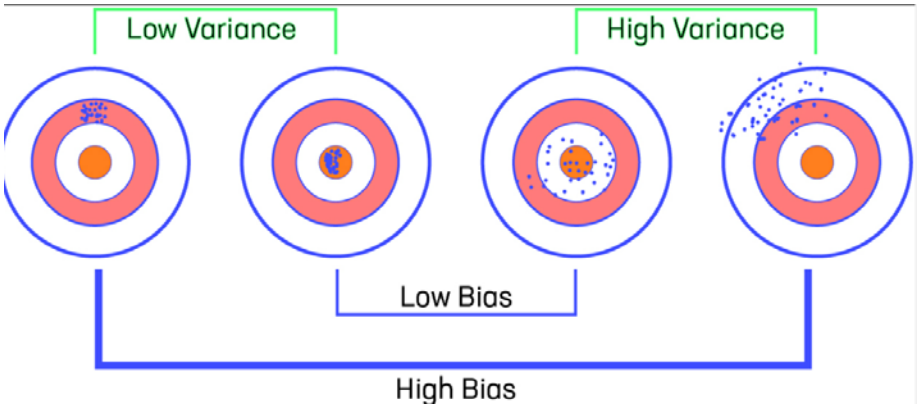
- ✓ here b is the batch size and m is number of training examples

Bias vs Variance

- ✓ **Bias** refers to the errors which occur when we try to fit a statistical model on real-world data which does not fit perfectly well on some mathematical model.
- ✓ **Variance** implies the error value that occurs when we try to make predictions by using data that is not previously seen by the model. There is a situation known as **high variance** that occurs when the model learns noise that is present in the data.



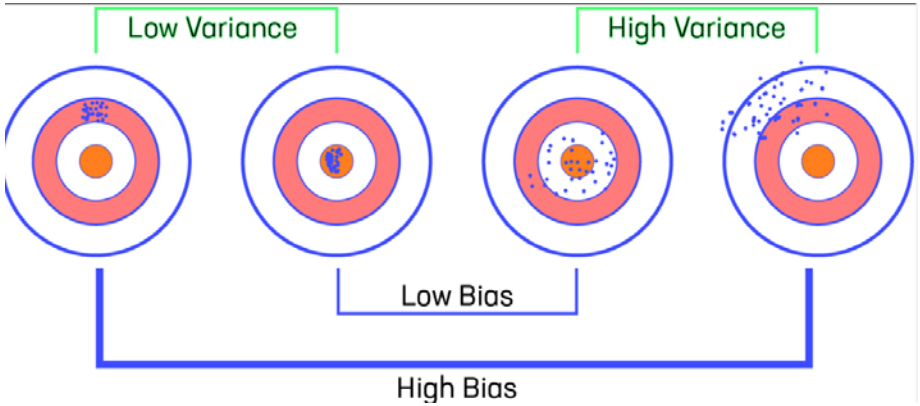
Bias vs Variance



- ✓ Let us consider that we have a very accurate model, this model has a **low error in predictions and it's not far from the target (which is represented by bull's eye). 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**. It works well for most of the classification/regression problems.

Bias vs Variance

- ✓ **Low Bias:** The average prediction is very close to the target value
- ✓ **High Bias:** The predictions differ too much from the actual value
- ✓ **Low Variance:** The data points are compact and do not vary much from their mean value
- ✓ **High Variance:** Scattered data points with huge variations from the mean value and other data points.
- ✓ To make a good fit, we need to have a correct balance of bias and variance.



What is Regularization?

- ✓ Regularization is a technique to **prevent the model from overfitting** by **adding extra information** to it.
- ✓ Sometimes the machine learning **model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data,** and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.
- ✓ It mainly **regularizes or reduces the coefficient of features toward zero**. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features."

How does Regularization Work?

- ✓ Regularization works by adding a penalty term to the cost function of a linear regression model, which reduces the magnitude of the coefficients or weights.
- ✓ By adding regularization, you can prevent the model from fitting too closely to the training data and reduce the variance of the model. However, regularization also introduces some bias to the model, as it shrinks the coefficients towards zero or a constant.
- ✓ Let's consider the simple linear regression equation:

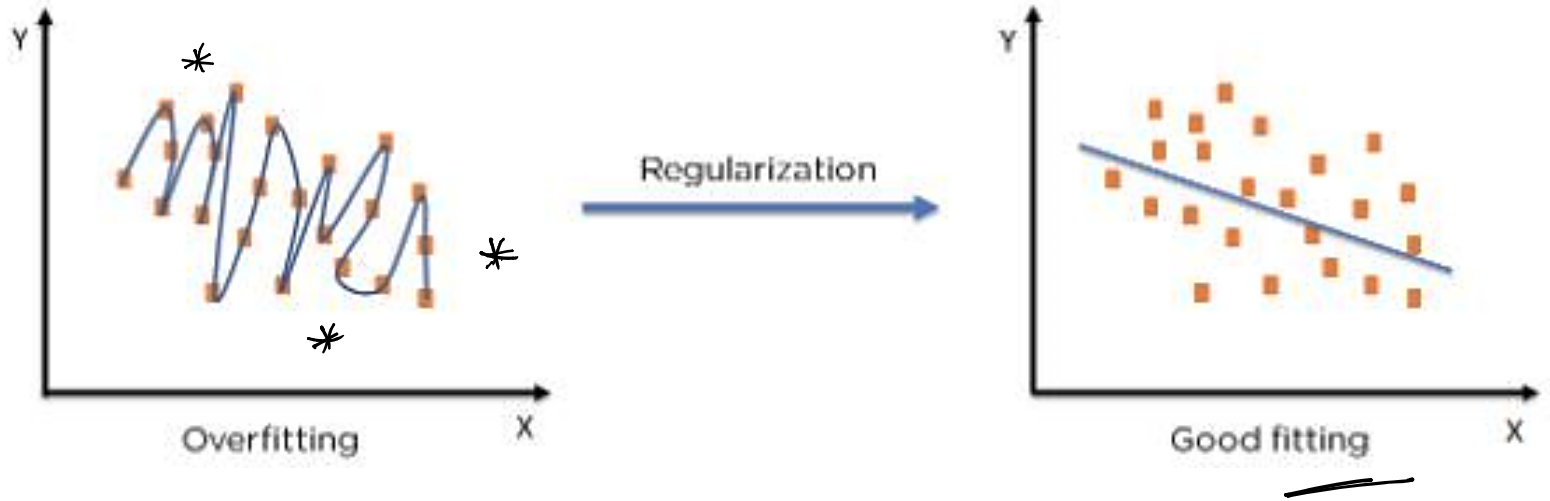
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n$$

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$

- ✓ In the above equation, Y represents the value to be predicted
- ✓ $X_1, X_2, \ldots X_n$ are the features for Y.
- ✓ $\beta_0, \beta_1, \ldots . \beta_n$ are the weights or magnitude attached to the features, respectively.
- ✓ Linear regression models try to optimize the coefficients to minimize the cost function. The equation for the cost function for the multiple linear model is given below:

$$\sum_{i=1}^{M} (y_i - y'_i)^2 = \sum_{i=1}^{M} (y_i - \sum_{j=0}^{n} \beta_j * Xij)^2$$

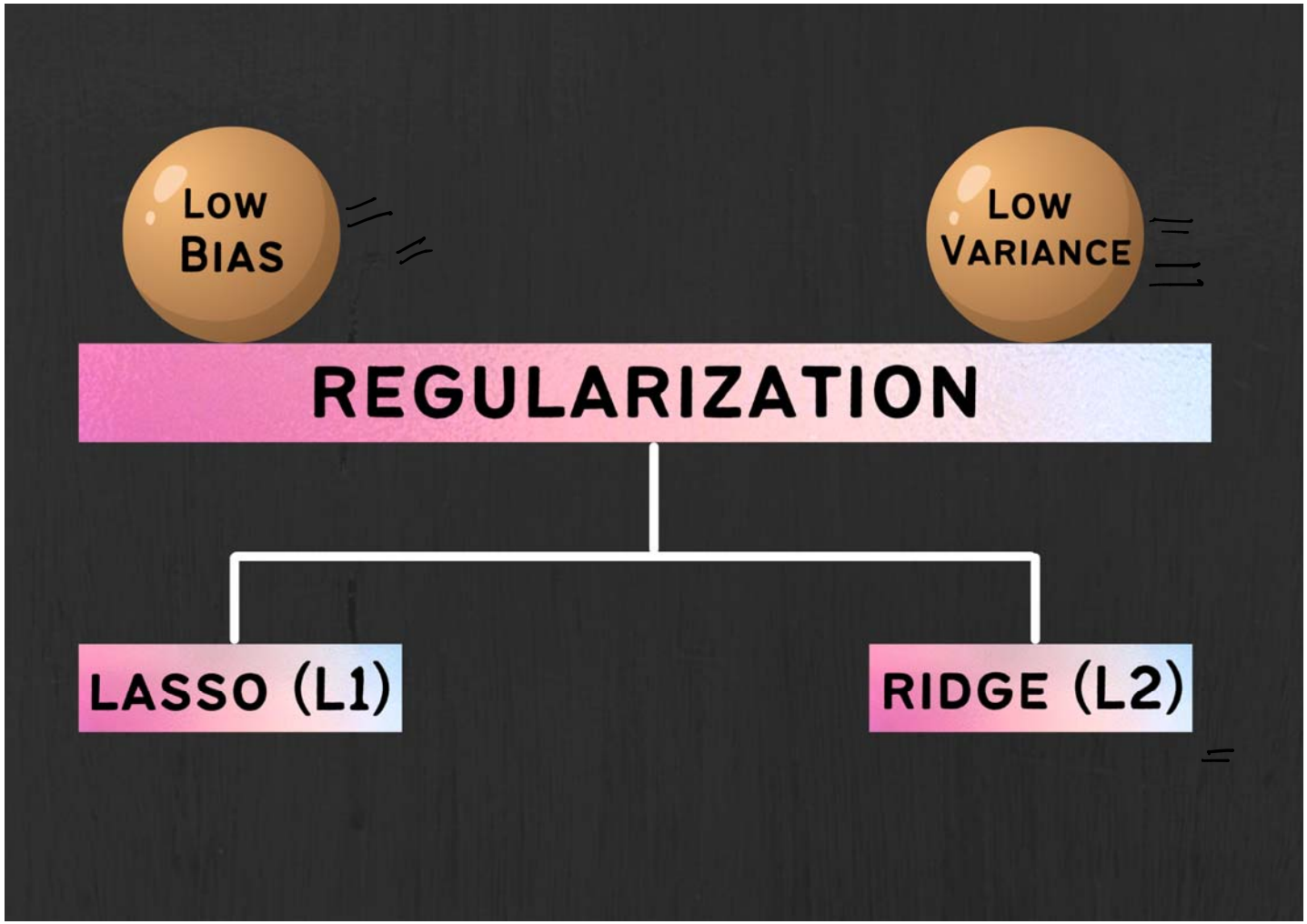
$\sum_{i=1}^{M} (\hat{y}_i - y_i) = \sum_{i=1}^{M} (\sum_{j=0}^{n} \theta_j x_{ij} - y_i)^2$



- ✓ Using Regularization, we can fit our machine learning model appropriately on a given test set and hence reduce the errors in it.

Techniques of Regularization

- ✓ There are mainly two types of regularization techniques, which are given below:
- ✓ **Ridge Regression**
- ✓ **Lasso Regression**


LOW BIAS
LOW VARIANCE
REGULARIZATION
LASSO (L1)
RIDGE (L2)

Ridge Regression

- ✓ Ridge regression is a **model tuning method** that is used to **analyze any data that suffers from multicollinearity**. When the issue of multicollinearity occurs, **least-squares are unbiased,** and variances are large, this results in **predicted values being far away from the actual values**. In modeled data, multicollinearity could be defined as the presence of a correlation between independent variables. Estimates of the regression coefficient may become inaccurate as a result.
- ✓ Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as **L2 regularization**.
- ✓ In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called **Ridge Regression penalty**. We can calculate it by **multiplying with the lambda to the squared weight of each individual feature.**

Ridge Regression

- ✓ The equation for the cost function in ridge regression will be:

$$\sum_{i=1}^{M}(y_i - y'_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n}\beta_j * x_{ij}\right)^2 + \lambda\sum_{j=0}^{n}\beta_j{}^2$$

- ✓ In the above equation, the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model.
- ✓ As we can see from the above equation, if the values of **λ tend to zero, the equation becomes the cost function of the linear regression model.** Hence, for the minimum value of λ, the model will resemble the linear regression model.
- ✓ A **general linear or polynomial regression will fail if there is high collinearity between the independent variables,** so to solve such problems, Ridge regression can be used.
- ✓ It helps to solve the problems **if we have more parameters than samples**.

Lasso Regression:

- ✓ Lasso regression is another regularization technique to reduce the complexity of the model. It stands for **Least Absolute Shrinkage and Selection Operator.**
- ✓ It is similar to the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights.
- ✓ Since it takes absolute values, hence, it **can shrink the slope to 0**, whereas **Ridge Regression can only shrink it near to 0.**
- ✓ It is also called as **L_1 regularization.** The equation for the cost function of Lasso regression will be:
- ✓ **Residual Sum of Squares + λ * (Sum of the absolute value of the magnitude of coefficients)**

$$\sum_{i=1}^{M}(y_i - y'_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n}\beta_j * x_{ij}\right)^2 + \lambda\sum_{j=0}^{n}|\beta_j|^{\square}$$

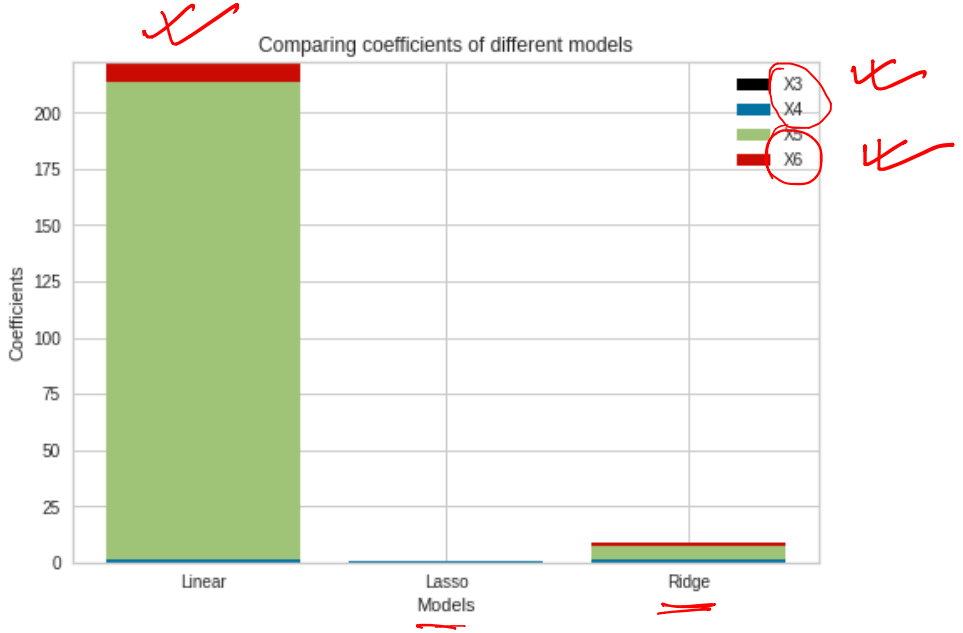
- ✓ Where, λ denotes the amount of shrinkage, controls the strength of the L_1 penalty
- ✓ λ = 0 implies all features are considered and it is equivalent to the linear regression where only the residual sum of squares is considered to build a predictive model
- ✓ λ = ∞ implies no feature is considered i.e, as λ closes to infinity it eliminates more and more features
- ✓ The bias increases with increase in λ
- ✓ variance increases with decrease in λ
- ✓ If an intercept is included in the model, it is usually left unchanged.

Lasso Regression

- ✓ LASSO regression encourages models with fewer parameters where some coefficients are forced to be exactly zero.
- ✓ This feature makes LASSO particularly useful for feature selection, as it can automatically identify and discard irrelevant or redundant variables.
- ✓ This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean.
- ✓ This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.
- ✓ Hence, the Lasso regression can help us to reduce the overfitting in the model as well as the feature selection.

Comparison of Theta Coefficients

- ✓ Inspecting the coefficients, we can see that Lasso and Ridge Regression had shrunk the coefficients, and thus the coefficients are close to zero. On the contrary, Linear Regression still has a substantial value of the coefficient for the X5 column.



Comparison of Theta Coefficients

Lasso Regression

Here's a step-by-step explanation of how LASSO regression works:

Linear regression model: LASSO regression starts with the standard linear regression model, which assumes a linear relationship between the independent variables (features) and the dependent variable (target). The linear regression equation can be represented as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \varepsilon$$ Where:

- y is the dependent variable (target).
- $\beta_0, \beta_1, \beta_2, ..., \beta_p$ are the coefficients (parameters) to be estimated.
- $x_1, x_2, ..., x_p$ are the independent variables (features).
- ε represents the error term.

L_1 regularization: LASSO regression introduces an additional penalty term based on the absolute values of the coefficients. The L_1 regularization term is the sum of the absolute values of the coefficients multiplied by a tuning parameter λ: $L_1 = \lambda * (|\beta_1| + |\beta_2| + ... + |\beta_p|)$ Where:

- λ is the regularization parameter that controls the amount of regularization applied.
- $\beta_1, \beta_2, ..., \beta_p$ are the coefficients.

Lasso Regression

Objective function: The objective of LASSO regression is to find the values of the coefficients that minimize the sum of the squared differences between the predicted values and the actual values, while also minimizing the L_1 regularization term: RSS + L_1 Where: RSS is the residual sum of squares, which measures the error between the predicted values and the actual values.

Shrinking coefficients: By adding the L_1 regularization term, LASSO regression can shrink the coefficients towards zero. When λ is sufficiently large, some coefficients are driven to exactly zero. This property of LASSO makes it useful for feature selection, as the variables with zero coefficients are effectively removed from the model.

Model fitting: To estimate the coefficients in LASSO regression, an optimization algorithm is used to minimize the objective function. Coordinate Descent is commonly employed, which iteratively updates each coefficient while holding the others fixed.

Key Difference between Ridge Regression and Lasso Regression

- ✓ **Ridge regression** is mostly used to reduce the overfitting in the model, and it includes all the features present in the model. It reduces the complexity of the model by shrinking the coefficients.
- ✓ **Lasso regression** helps to reduce the overfitting in the model as well as feature selection.

What is Multicollinearity?

- ✓ In modeled data, multicollinearity could be defined as the presence of a correlation between independent variables. Estimates of the regression coefficient may become inaccurate as a result.

Example: Multicollinearity

	Dep.	Ind.1	Ind.2	Ind.3
Dep.	1.00			
Ind.1	.72	1.00		
Ind.2	.67	.75	1.00	
Ind.3	.60	.85	.85	1.00

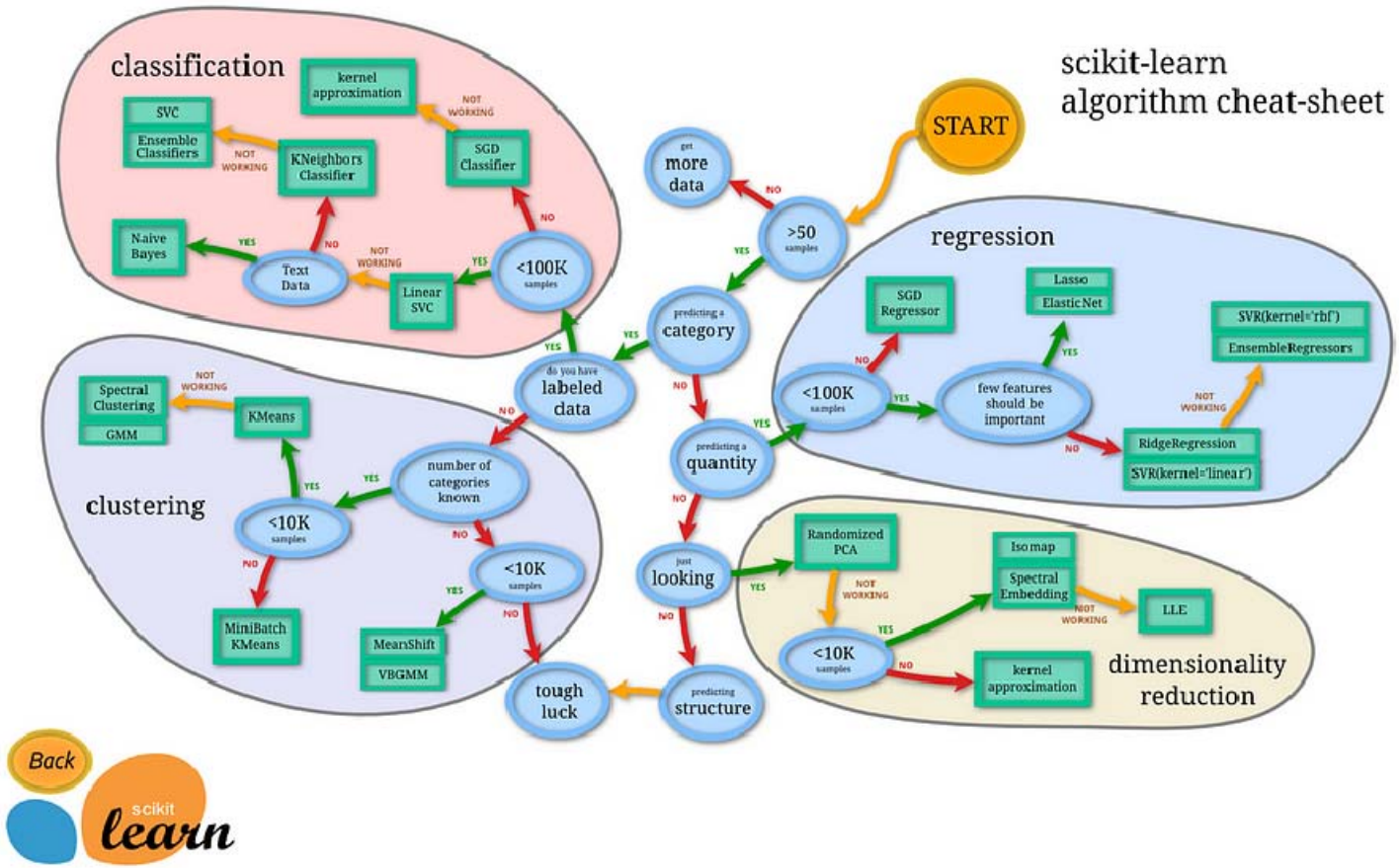
How λ relates to the principle of "Curse of Dimensionality"?

- ✓ As the value of λ rises, it significantly reduces the value of coefficient estimates and thus reduces the variance. Till a point, this increase in λ is beneficial for our model as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data. But after a certain value of λ, the model starts losing some important properties, giving rise to bias in the model and thus underfitting. Therefore, we have to select the value of λ carefully. To select the good value of λ, cross-validation comes in handy.

Important points about λ:

- ✓ λ is the tuning parameter used in regularization that decides how much we want to penalize the flexibility of our model i.e, **controls the impact on bias and variance**.
- ✓ When **λ = 0**, the penalty term has no effect, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of λ i.e, λ=0, the model will resemble the linear regression model. So, the estimates produced by ridge regression will be equal to least squares.
- ✓ However, as **λ→∞** (tends to infinity), the impact of the shrinkage penalty increases, and the ridge regression coefficient estimates will approach zero.

Thank You


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
YES
Linear SVC
<100K samples
NO
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
RidgeRegression
NO
SVR(kernel='linear')
predicting a category
YES
NO
do you have labeled data
YES
NO
predicting a quantity
YES
NO
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
clustering
<10K samples
NO
NO
<10K samples
YES
MiniBatch KMeans
MeanShift
VBGMM
NO
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure
Back
scikit learn

Loss function for multiple linear regression

MSE, $L = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$

For m samples

x_1	x_2	y
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2
x_{31}	x_{32}	y_3

$= \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \cdots \theta_n x_{in} - y_i)^2$ For n Features

$= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(\quad)^2$

Now for simplicity For 2 Features and for $m = 2$ samples

$L = \frac{1}{2}\left[(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)^2 + (\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2\right]$

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

$\hat{y}_1 = \theta_0 + \theta_1 x_{11} + \theta_2 x_{12}$

$\hat{y}_2 = \theta_0 + \theta_1 x_{21} + \theta_2 x_{22}$

Sub:

Day / Date / Time:

x_1	x_2	y
8.1	9.3	9.2
9.5	95	3.5

Again,

$L = \frac{1}{2}\cdot\frac{1}{2}\left[(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)^2 + (\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2\right]$

$\frac{\delta L}{\delta \theta_0} = \frac{1}{2}\cdot\frac{1}{2}\left[2(\theta_0 + \theta_1 x_{11} + \theta_2 x_{12} - y_1)(1) + 2(\theta_0 + \theta_1 x_{21} + \theta_2 x_{22} - y_2)(1)\right]$

$= \frac{1}{2}\cdot\frac{1}{2}\left[2(\hat{y}_1 - y_1) + 2(\hat{y}_2 - y_2)\right]$

$= \frac{1}{2}\cdot\frac{2}{2}\left[(\hat{y}_1 - y_1) + (\hat{y}_2 - y_2)\right]$

Now for m samples

$\frac{\delta L}{\delta \theta_0} = \frac{1}{2}\cdot\frac{2}{m}\left[(\hat{y}_1 - y_1) + (\hat{y}_2 - y_2) + (\hat{y}_3 - y_3) + \cdots + (\hat{y}_m - y_m)\right]$

$\frac{\delta m}{\delta \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)$ ✓

$$\theta_0 = \theta_0 - \alpha \frac{\delta L}{\delta \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\delta L}{\delta \theta_1}$$

$$\theta_2 = \theta_2 - \alpha \frac{\delta L}{\delta \theta_2}$$

$$\theta_j = \theta_j - \alpha \frac{\delta L}{\delta \theta_j} \quad [j = 0, 1, 2, \cdots n]$$

Again,

$$L = \frac{1}{2} \cdot \frac{1}{2} \sum_{i=1}^{m=2} (\hat{y}_i - y_i)^2$$

$$\sum_{i=1}^{M} (y_i - y'_i)^2 = \sum_{i=1}^{M} (y_i - \sum_{j=0}^{n} \beta_j * Xij)^2$$

$$= \frac{1}{2} \cdot \frac{1}{2} \left[(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \right]$$

$$= \frac{1}{2} \cdot \frac{1}{2} \left[(\theta_0 - \theta_1 x_{11} - \theta_2 x_{12} - y_1)^2 + (\theta_0 - \theta_1 x_{21} + \theta_2 x_{22} - y_2)^2 \right]$$

$$\frac{\delta L}{\delta \theta_1} = \frac{1}{2} \cdot \frac{1}{2} \left[2 (\hat{y}_1 - y_1)(x_{11}) + 2 (\hat{y}_2 - y_2)(x_{21}) \right]$$

11, 21, 31, ... m1

$$\frac{\delta L}{\delta \theta_1} = \frac{1}{2} \cdot \frac{2}{m} \left[(\hat{y}_1 - y_1)(x_{11}) + (\hat{y}_2 - y_2)(x_{21}) + (\hat{y}_3 - y_3)(x_{31}) + \cdots + (\hat{y}_n - y_n)(x_{m1}) \right]$$

$$\therefore \frac{\delta L}{\delta \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{i1})$$

$\theta_1 \rightarrow$ Value of 1st-

$$\therefore \frac{\delta L}{\delta \theta_2} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)(x_{i2})$$

$$\therefore \frac{\delta L}{\delta \theta} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{in}$$ — column

x_1	x_2	y
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2
x_{31}	x_{32}	y_3
...	...	...
x_{m1}	x_{m2}	y_m

x_{i1} means

$$\begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \vdots \\ x_{m1} \end{bmatrix}$$ 1st- column

Now, $\theta_j = \theta_j - \alpha \frac{\delta L}{\delta \theta_j}$ where, $j = 0, 1, 2, \cdots n$

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{ij}$$

Finally,

Repeat until convergence

{

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{ij}$$

}

where $j = 0, 1, 2, \cdots n$

x_1	x_2	...	x_n	y
x_{11}	x_{12}	...	x_{1n}	y_1
x_{21}	x_{22}	...	x_{2n}	y_2
x_{31}	x_{32}	...	x_{3n}	y_3
...	...	...	...	...
x_{m1}	x_{m2}	...	x_{mn}	y_m

or we can write,

Repeat until convergence

{

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{ij}$$

}

$\theta_j = 0, 1, 2, \cdots n$

{ Repeat until convergence

$$\theta_0 = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 = \theta_1 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i1}$$

$$\theta_2 = \theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i2}$$

$$\vdots$$

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{in}$$

}

Now, update $\theta_1, \theta_2 \cdots$

Steps 1: Taking random values, $\theta_0 = 0$, $\theta_1 = 1$, $\theta_2 = 1$.

Steps 2: epoch = 100, $\alpha = 0.1$. [for 2 rows and 2 Column
$m = 2$
$n = 2$]

Algorithm

Below is the algorithm for the Least Square Solution using Coordinate Descent.
Notice, $r + x_j w_j$ is nothing but $(y_i - \sum_{k \neq j}^{p} x_{ik} w_k)$.

- Normalize X
- Initialize `w` with zeros (or randomly)
- LOOP O: Until Convergence
  - $r := \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij} w_j \right)$
  - LOOP P: For j = 1, 2, ... p
    - $r_{-j} = r + x_j w_j$
    - $w_j = \sum_{i=1}^{N} x_{ij} r_{-j}$
    - $r = r - x_j w_j$