LINEAR MODEL - LINEAR REGRESSION

Dr. Debanjan Pathak

Formulation & Mathematical Foundation of Regression Problem

What is **Regression**

- Regression – predict value of *response* **variable (Y)**from attribute **variables (X)**.
- Variables – continuous numeric values
- Regression analysis – a set of **statistical processes** for estimating the relationships between a dependent variable and one or more independent variables.
  - Dependent variables are often called the 'predictand', 'outcome' or 'response' variable;
  - Independent variables are often called 'predictors', 'covariates', 'explanatory variables' or 'features'.
  - Regression analysis is a way of mathematically sorting out which of those variables does indeed have an impact.
  - Used for modeling the future relationship between the variables.
- Statistical process – a **science** of collecting, exploring, organizing, analyzing, interpreting data and exploring patterns and trends to answer questions and make decisions (Broad area).

Basics of **Regression Models**

- Regression models predict a value of the Y variable given known values of the X variables.
- Prediction within the range of values in the dataset used for model-fitting is known as interpolation.
- Prediction outside this range of the data is known as extrapolation.
- First, a model to estimate the outcome need to be fixed.
- Then the parameters of that model need to be estimated using any chosen method (e.g., least squares).

Formulation of **Regression Models**

- Regression models involve the following components:
- The unknown parameters, often denoted as $\beta \; or \; \omega \; or \; w$.
- The independent variables, which are observed in data and are often denoted as a vector X_i (where i denotes a row of data).
- The dependent variable, which are observed in data and often denoted using the scalar Y_i.
- The error terms, which are not directly observed in data and are often denoted using the scalar e_i.

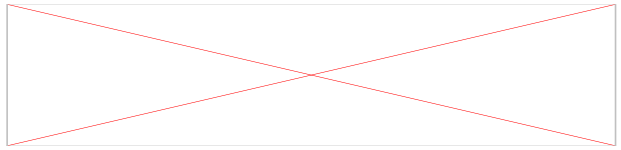
Formulation of **Regression Models**

- Most regression models propose that Y_i is a function of X_i and β, with e_i representing an additive error term that may stand in for a random statistical noise.

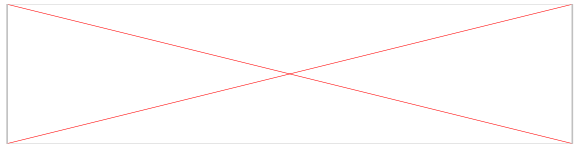
- Our objective is to estimate the function $f(X_i , \beta)$ that most closely fits the data.
- To carry out regression analysis, the form of the function f must be specified.
- Sometimes the form of this function is based on knowledge about the relationship between Y_i and X_i .
- If no such knowledge is available, a flexible or convenient form for f is chosen.

Formulation of **Regression Models**

- You may start with a simple univariate linear regression:



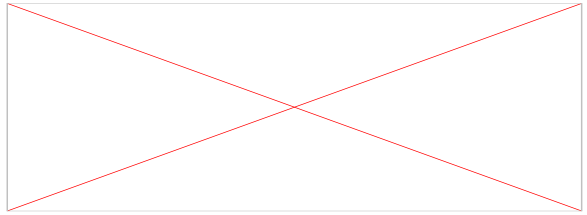
- It indicates that you believe that a reasonable approximation for Y_i is:



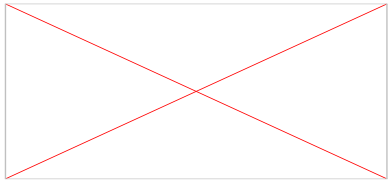
- Now the next objective is to estimate the parameters β
  - may be using least squares method
  - may go with other alternatives such as least absolute deviations, Least trimmed squares, quantile regression estimator, Theil–Sen estimator, M-estimation (maximum likelihood type) or S-estimation (scale).

Formulation of **Regression Models**

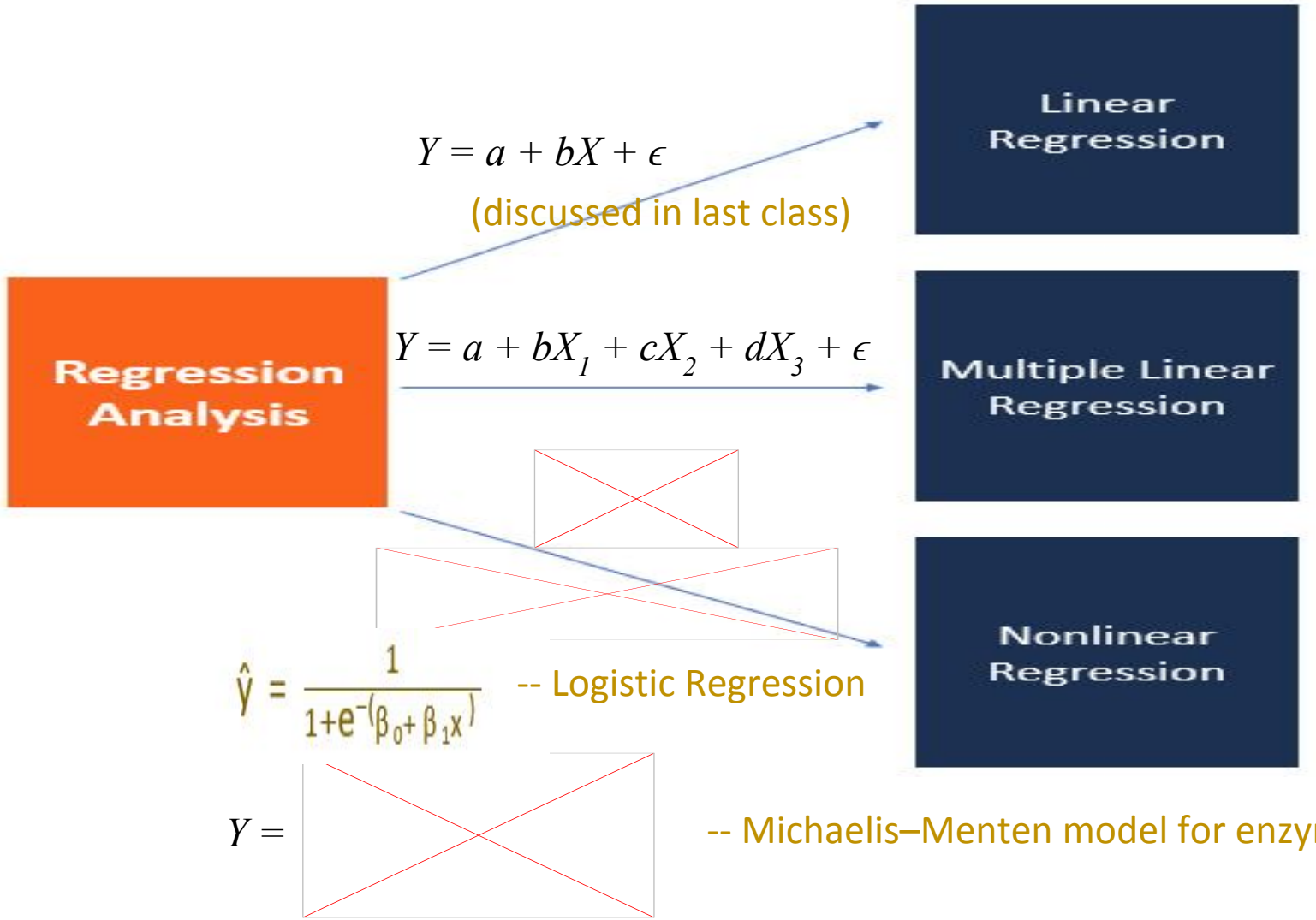
- Find the value of β that minimizes the sum of squared errors



- A given regression method will ultimately provide an estimate of β, usually denoted $\widehat{\beta}$.
- Using this estimate, you can then find the fitted value for prediction or to assess the accuracy of the model in explaining the data.



Variants in **Regression Models**



- The most common models are simple linear and multiple linear.
- Nonlinear regression analysis is commonly used for more complicated data sets in which the dependent and independent variables show a nonlinear relationship.

Multiple Linear Regression

		x			y
i	j →	**Size (feet^2)** 1	**Number of bedrooms** 2	**Number of floors** 3	**Age of home (years)** 4	**Price ($1000)**
1		2104	5	1	45	460
2		1416	3	2	40	232
3		1534	3	2	30	315
4		852	2	1	36	178
N		…	…	…	…	…

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

N = number of training examples

Multiple Linear Regression

$$\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + \ldots + w_n x_n^{(i)}; i = 1, \ldots, N$$

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

N = number of training examples

Use the following data set to answer the questions:

Sample	Y	X1	X2	X3
1	35	8.4	8	1
2	10	2	6.5	8.5
3	9	3.5	6.2	6.5
4	30	10.4	5	1.5
5	20	6.5	6.5	7.5
6	23	6.2	7.3	4.5
7	28	12.4	6.4	4
8	8	7	6	10
9	29	5.8	6.1	3
10	4	3	5.4	11
11	18	6	7.3	4.5
12	14	5.5	6.6	5.5
13	32	9	6.5	2.5
14	6	1.1	5.8	7
15	8	2.1	7.1	9
16	37	10	8.5	2
17	25	7	5.5	3
18	15	5	5	4.5
19	30	9.3	7.9	3
20	10	4.4	4.5	7.9

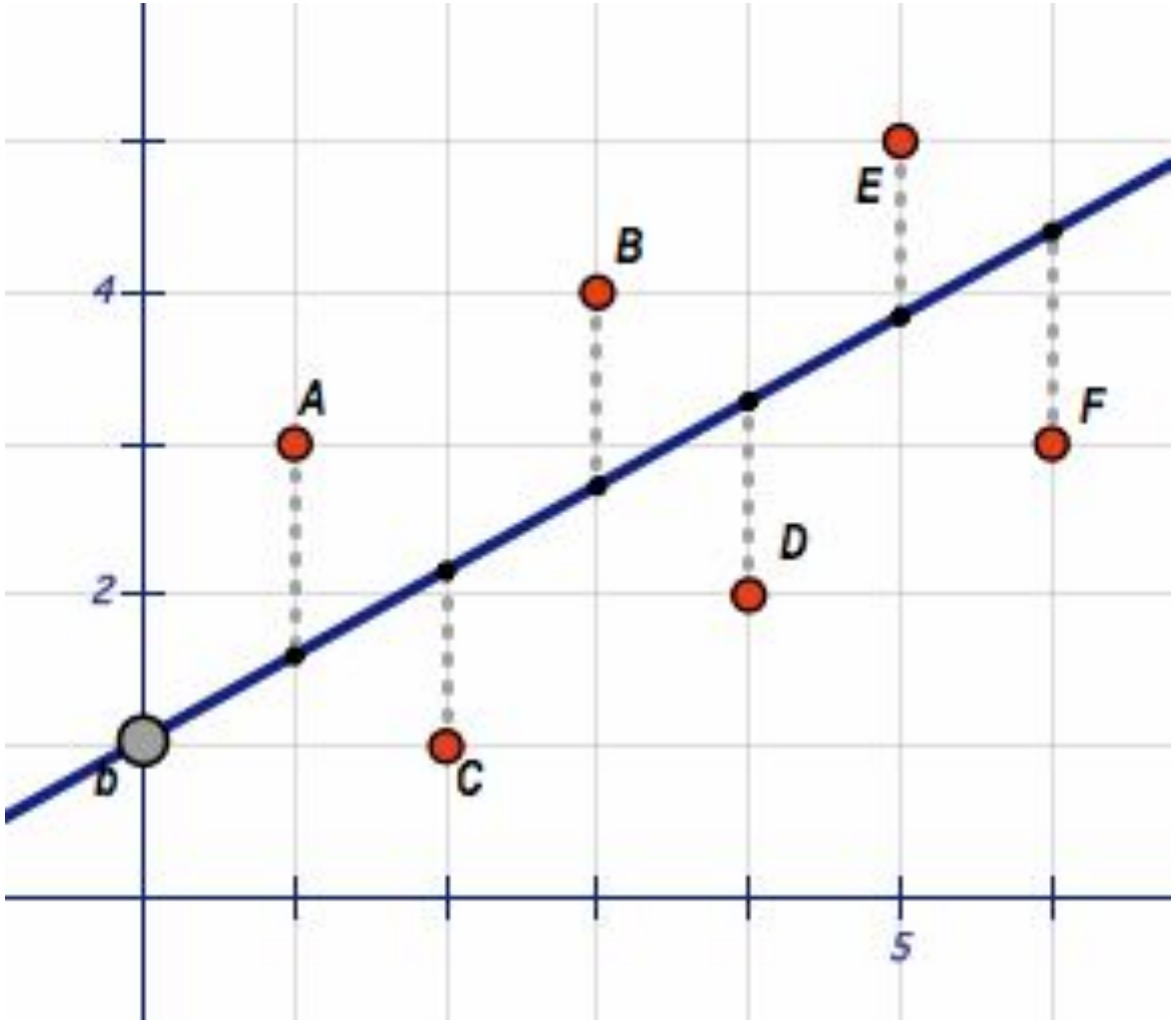
We are interested in modeling Y based on X1, X2 and X3.
Y=length of larvae in the water
X1: depth of the water
X2: amount of dissolved oxygen
X3: brackishness of water.

The Regression Model, & The Concepts of Least Squares

What is **Least Square Method**

- The least-squares method is a statistical method that is practised to find a regression line or a best-fit line for the given pattern.
- The method of least squares is used in regression.
- In regression analysis, this method is said to be a standard approach for the approximation of sets of equations having more equations than the number of unknowns (overdetermined systems).
- It is used to approximate the solution by minimizing the sum of the squares of the residuals made in the results of each individual equation.
  - Residual: the difference between an observed value and the fitted value provided by a model
- The problem of finding a linear regressor function will be formulated as a problem of minimizing a criterion function.
- The widely-used criterion function for regression purposes is the

Least Square Method with Linear Regression



Least Square Method with Linear Regression

- In general, regression methods are used to predict the value of response (dependent) variable from attribute (independent) variables,
- Linear regressor model fits a linear function (relationship) between dependent (output) variable and independent (input) variables.

$$\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)}; i = 1, \ldots, N$$

  - where $\{w_0, w_1, \ldots, w_n\}$ are the parameters of the model.
- The method of linear regression is to choose the (n + 1) coefficients $w_0, w_1, \ldots, w_n$, to minimize the residual sum of squares of these differences over all the N training instances.

Least Square Method with Linear Regression

Residual error

$$e^{(i)} = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - \sum_{j=0}^{n} w_j x_j^{(i)}; x_0^{(i)} = 1$$

Residual sum-of-error-squares

$$E = \sum_{i=1}^{N} (e^{(i)})^2 = \sum_{i=1}^{N} \left(y^{(i)} - \sum_{j=0}^{n} w_j x_j^{(i)} \right)^2; x_0^{(i)} = 1$$

- The method of linear regression is to choose the (n + 1) coefficients w_0, w_1, …, w_n, to minimize the residual sum of squares of these differences over all the N training instances.

Minimal Sum-of-Error-Squares

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(N)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(N)} \\ \vdots & \vdots & & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(N)} \end{bmatrix} = [\overline{\mathbf{x}}^{(1)}\ \overline{\mathbf{x}}^{(2)} \cdots \overline{\mathbf{x}}^{(N)}]$$

$$\mathbf{y} = [y^{(1)}\ \ y^{(2)} \cdots y^{(N)}]^T$$

$$\overline{\mathbf{w}} = [w_0\ \ w_1\ \ w_2 \cdots w_n]^T$$

$$\overline{\mathbf{x}} = [1\ \ x_1\ \ x_2 \cdots x_n]^T$$

- For an optimum solution for w , the following equations need to be satisfied:

$$y^{(1)} = \overline{\mathbf{w}}^T\, \overline{\mathbf{x}}^{(1)}$$

$$\vdots$$

$$y^{(N)} = \overline{\mathbf{w}}^T\, \overline{\mathbf{x}}^{(N)}$$

Minimal Sum-of-Error-Squares

Therefore,

$$[y^{(1)}\ y^{(2)} \cdots y^{(N)}] = \bar{\mathbf{w}}^T [\bar{\mathbf{x}}^{(1)}\ \bar{\mathbf{x}}^{(2)} \cdots \bar{\mathbf{x}}^{(N)}] = \bar{\mathbf{w}}^T \mathbf{X}$$

or

$$\mathbf{y} = (\bar{\mathbf{w}}^T \mathbf{X})^T$$

The vector of residual errors becomes

$$\mathbf{y} - (\bar{\mathbf{w}}^T \mathbf{X})^T$$

Hence the error function can be written as,

$$E(\bar{\mathbf{w}}) = [\mathbf{y} - (\bar{\mathbf{w}}^T \mathbf{X})^T]^T\ [\mathbf{y} - (\bar{\mathbf{w}}^T \mathbf{X})^T]$$

$$= \bar{\mathbf{w}}^T [\mathbf{X}\mathbf{X}^T]\, \bar{\mathbf{w}} - 2\bar{\mathbf{w}}^T\ \mathbf{X}\mathbf{y} + \mathbf{y}^T\mathbf{y}$$

Minimal Sum-of-Error-Squares

$$E(\bar{\mathbf{w}}) = \bar{\mathbf{w}}^T [\mathbf{X}\mathbf{X}^T] \bar{\mathbf{w}} - 2\bar{\mathbf{w}}^T \mathbf{X}\mathbf{y} + \mathbf{y}^T\mathbf{y}$$

- In this least-squares estimation task, the objective is to find the optimal $\bar{w}$* that minimizes E ($\bar{w}$).
- The solution to this classic problem in calculus is found by setting the gradient of E ($\bar{w}$), with respect to $\bar{w}$, to zero.

$$\frac{\partial E(\bar{\mathbf{w}})}{\partial \bar{\mathbf{w}}} = 2(\mathbf{X}\mathbf{X}^T)\,\bar{\mathbf{w}} - 2\mathbf{X}\mathbf{y} = \mathbf{0}$$

This gives

$$\bar{\mathbf{w}}^* = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}$$

The fitted output values at the training data are

$$\hat{\mathbf{y}} = \mathbf{X}^T\,\bar{\mathbf{w}}^* = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}$$

Minimal Sum-of-Error-Squares

$$\hat{\mathbf{y}} = \mathbf{X}^T \bar{\mathbf{w}}^* = \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X}\mathbf{y}$$

- The (n + 1) x N matrix $X^+ = (XX^T)^{-1}X$ is called the pseudoinverse matrix of the matrix X^T. Thus, the optimal solution is

$$\bar{w}^* = X^+ y$$

Unique solution?

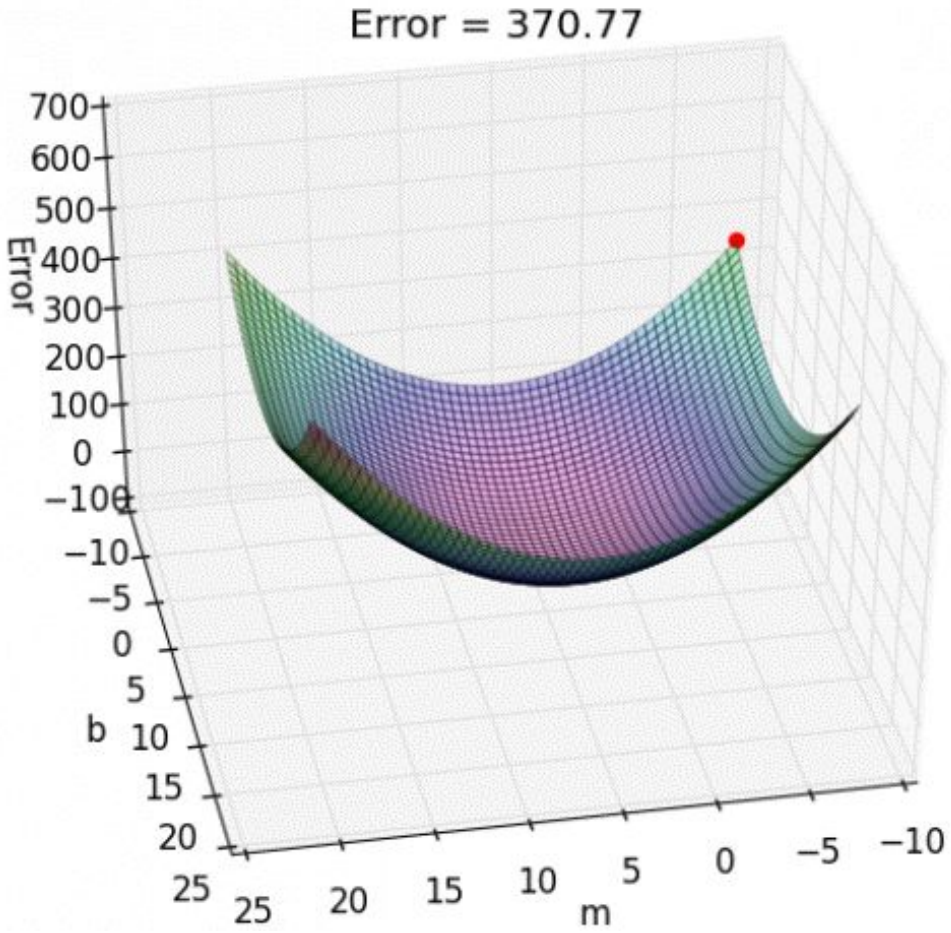
- It might happen that the columns of X are not linearly independent.
- Then XX^T is singular and the least squares coefficients $\overline{w}*$ are not uniquely defined.
- The singular case occurs most often when two or more inputs were perfectly correlated.
- A natural way to resolve the non-unique representation is by dropping redundant columns in X.
- Most regression software packages detect these redundancies and automatically implement some strategy for removing them.

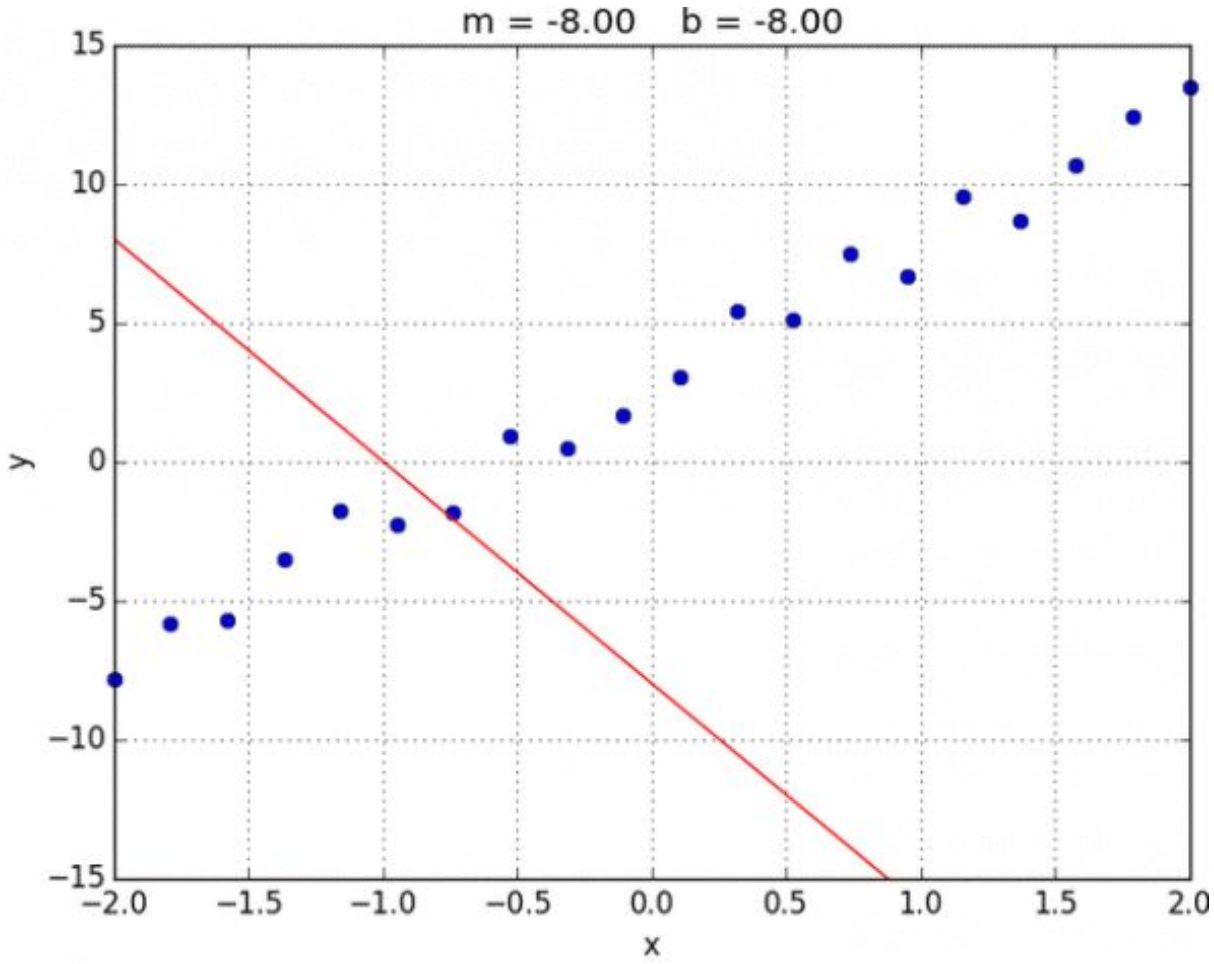
Error Reduction-Gradient Descent

Basics of **Gradient Descent**

- Gradient descent search helps determine a weight vector that minimizes E by starting with an arbitrary initial weight vector and then altering it again and again in small steps.
- Batch gradient descent: When the weight update is calculated based on all examples in the training dataset, it is called as batch gradient descent.
- Stochastic gradient descent: When the weight update is calculated incrementally after each training example or a small group of training example, it is called as stochastic gradient descent.
- Gradient descent procedure has two advantages over merely computing the pseudoinverse:
- (1) it avoids the problems that arise when XX^T is singular (it always yields a solution regardless of whether or not XX^T is singular);
- (2) it avoids the need for working with large matrices.

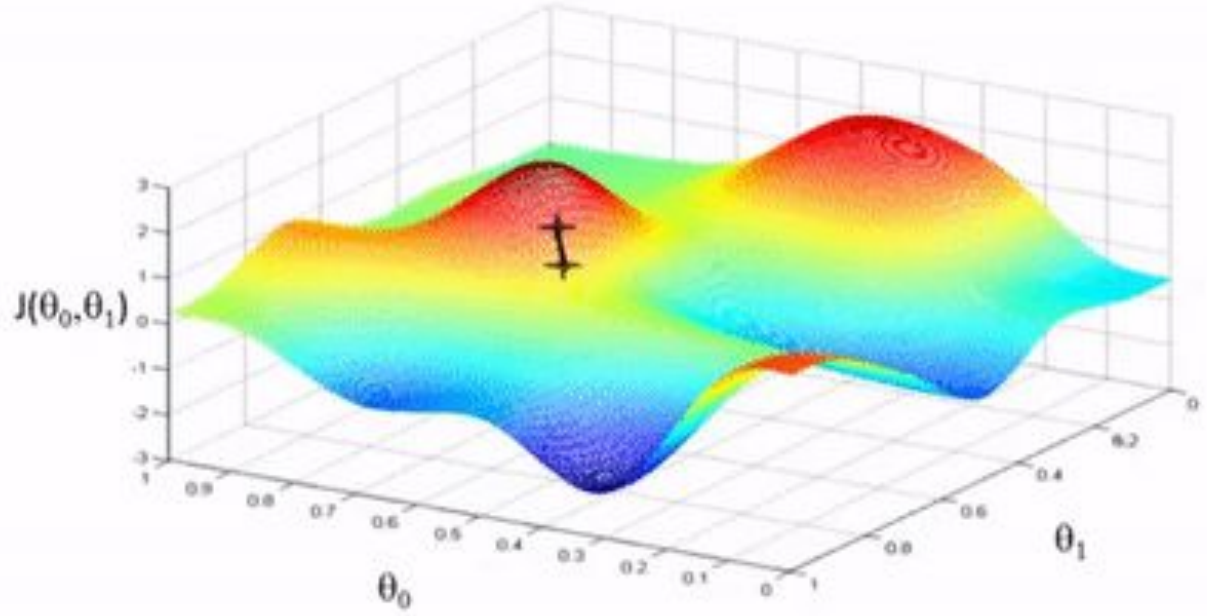
Basics of **Gradient Descent**

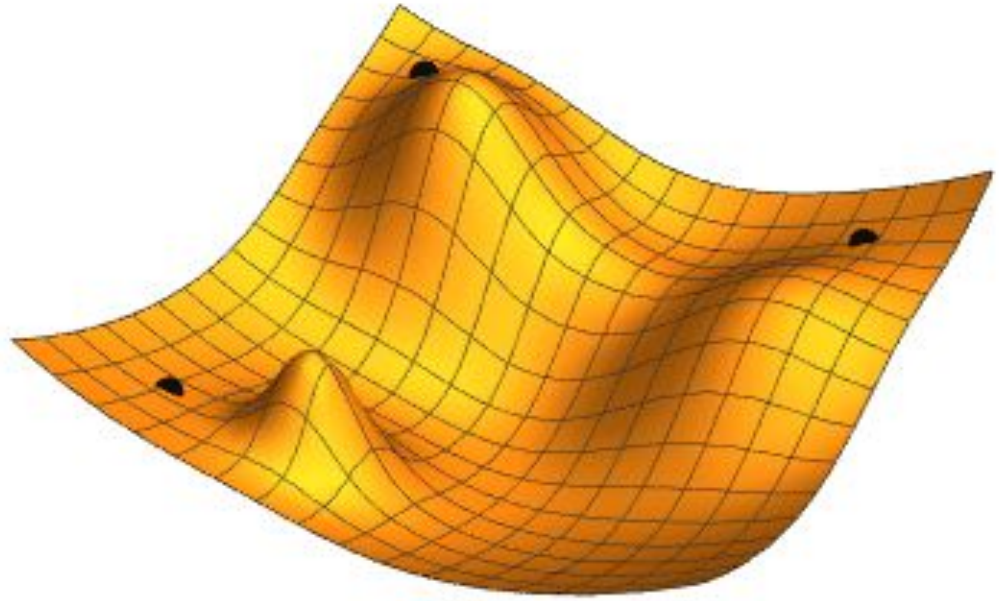




Basics of **Gradient Descent**

- The error surface may have multiple local minimums but a single global minimum.
- The objective would be to find out global minimum.





What is **Gradient Descent**

Gradient Descent Optimization Schemes

- Optimization method *Gradient Descent Method* used for minimization tasks. Changes of the weights are made according to the following algorithm:

$$\bar{\mathbf{w}}_{k+1} = \bar{\mathbf{w}}_k - \eta \frac{\partial E}{\partial \bar{\mathbf{w}}} |_k$$

where η denotes the *learning rate*, and k stands for the actual iteration step.

Note:

- Need to choose η.
- Needs many iterations.
- Works well even when n is large.
- Gradient descent serves as the basis for learning algorithms that search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.

What is **Gradient Descent**

Gradient Descent Optimization Schemes

- Optimization method *Gradient Descent Method* used for minimization tasks. Changes of the weights are made according to the following algorithm:

$$\bar{\mathbf{w}}_{k+1} = \bar{\mathbf{w}}_k - \eta \frac{\partial E}{\partial \bar{\mathbf{w}}} |_k$$

where η denotes the *learning rate*, and k stands for the actual iteration step.

Approaches for deciding the iteration step:

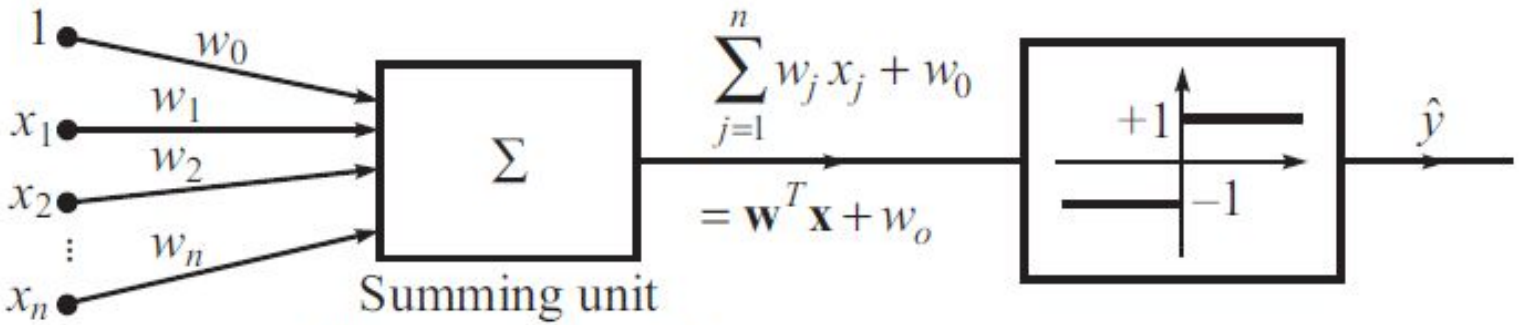
1. Batch methods use all the data in one shot.

- k^{th} iteration step means the k^{th} presentation of training dataset.
- Gradient is calculated across the entire set of training patterns.

2. Online methods is where

- k – iteration step after single data pair is presented.
- Share almost all good features of recursive least square algorithm with reduced computational complexity.

The **gradient descent** training rule



Linear classification using regression technique

$$g(\mathbf{x}) = \sum_{j=1}^{n} w_j x_j + w_0 = \mathbf{w}^T\mathbf{x} + w_0$$

Let us define a single weight vector $\bar{\mathbf{w}}$ for the weights $(\mathbf{w}, w_0)$:

$$\bar{\mathbf{w}}^T = [w_0\ w_1\ w_2\ \ldots\ w_n]^T$$

In terms of the weight vector $\bar{\mathbf{w}}$, the output

$$g(\mathbf{x}) = \bar{\mathbf{w}}^T\bar{\mathbf{x}}$$

where

$$\bar{\mathbf{x}}^T = [x_0\ x_1\ x_2\ \ldots\ x_n]^T;\ x_0 = 1$$

Performance Criterion:

$$E(\bar{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \bar{\mathbf{w}}^T\bar{\mathbf{x}}^{(i)})^2$$

---To be minimized.
Called as cost function.

used for computational convenience

The **gradient descent** training rule

- The error surface is parabolic with a single global minimum.
- The specific parabola will depend on the particular set of training examples.
- The direction of steepest descent along the error surface can be found by computing the derivative of E with respect to each component of the vector $\bar{w}$.
- This vector-derivative is called the gradient of E w.r.t $\bar{w}$, written $\nabla E(\bar{w})$

$$\nabla E(\bar{\mathbf{w}}) = \left[\frac{\partial E}{\partial w_0} \; \frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_n} \right]^T$$

Remember, it can be applied to any objective function, not just for squared distances.

- The negative of this vector gives the direction of steepest decrease.
- Therefore, the training rule for gradient descent is,

$$\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} + \Delta\bar{\mathbf{w}}$$

$$\Delta\bar{\mathbf{w}} = -\eta \nabla E(\bar{\mathbf{w}})$$

Performance Criterion:

$$E(\bar{\mathbf{w}}) = \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - \bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(i)})^2$$

---To be minimized.
Called as cost function.

here, η *is learning rate, a +ve constant, determines the step size in the search.*

The **gradient descent** training rule

- This training rule can also be written in its component form:

$$w_j \leftarrow w_j + \Delta w_j; j = 0, 1, 2, \ldots, n$$

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j}$$

- which shows that steepest descent is achieved by altering each component w_j of w in proportion to $\frac{\partial E}{\partial w_j}$
- starting with an arbitrary initial weight vector, is changed in the direction producing the steepest descent along the error surface.
- The process goes on till the global minimum error is attained.

Performance Criterion:

$$E(\overline{\mathbf{w}}) = \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - \overline{\mathbf{w}}^T \overline{\mathbf{x}}^{(i)})^2$$

---To be minimized.
Called as cost function.

The **gradient** with respect to weight w_j

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j x_j^{(i)} + w_0\right)\right)^2\right]$$

The error $e^{(i)}$ for the i^{th} sample of data is given by $e^{(i)} = y^{(i)} - \left(\sum_{j=1}^{n} w_j x_j^{(i)} + w_0\right)$.
It follows that

$$\frac{1}{2}\sum_{i=1}^{N}\left[\frac{\partial}{\partial w_j}(e^{(i)})^2\right] = \sum_{i=1}^{N} e^{(i)}\frac{\partial e^{(i)}}{\partial w_j}$$

& $$\frac{\partial e^{(i)}}{\partial w_j} = -x_j^{(i)}$$

→ $$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{N}(e^{(i)})^2\right]$$

$$= -\sum_{i=1}^{N} e^{(i)} x_j^{(i)}$$

$$= -\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j x_j^{(i)} + w_0\right)\right)x_j^{(i)}$$

Performance Criterion:

$$E(\overline{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \overline{\mathbf{w}}^T\overline{\mathbf{x}}^{(i)})^2$$

---To be minimized.
Called as cost function.

The **gradient** with respect to weight w_j

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{\quad}^{n} \quad\right)\right)^2\right]$$

The error $e^{(i)}$ for the i^{th} sample of data is given by e

It follows that

$$\frac{1}{2}\sum_{i=1}^{N}\left[\frac{\partial}{\partial w_j}(e^{(i)})^2\right] = \sum_{i=1}^{N} e^{(i)}\,\frac{\partial e^{(i)}}{\partial w_j}$$

&

$$\frac{\partial e^{(i)}}{\partial w_j} = -\,x_j^{(i)}$$

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{N}(e^{(\quad)})^2\right.$$

$$= -\sum_{i=1}^{N} e^{(i)}\,x_j^{(i)}$$

$$= -\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j\,x_j^{(i)} + w_0\right)\right)x_j^{(i)}$$

The gradient with respect to bias,

$$\frac{\partial E}{\partial w_0} = -\sum_{i=1}^{N} e^{(i)} = -\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j\,x_j^{(i)} + w_0\right)\right)$$

Therefore, the weight update rule for gradient descent becomes

$$w_j \leftarrow w_j + \eta\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j\,x_j^{(i)} + w_0\right)\right)x_j^{(i)}$$

$$w_0 \leftarrow w_0 + \eta\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j\,x_j^{(i)} + w_0\right)\right)$$

An epoch is a complete run through all the N associated pairs.

The **gradient** with respect to weight w_j

The gradient with respect to bias,

$$\frac{\partial E}{\partial w_0} = -\sum_{i=1}^{N} e^{(i)} = -\sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j x_j^{(i)} + w_0\right)\right)$$

Therefore, the weight update rule for gradient descent becomes

$$w_j \leftarrow w_j + \eta \sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j x_j^{(i)} + w_0\right)\right) x_j^{(i)}$$

$$w_0 \leftarrow w_0 + \eta \sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n}\right.\right.$$

- **Once an epoch is completed, the pair ($x^{(1)}$, $y^{(1)}$) is presented again and a run is performed through all the pairs again.**
- **After several epochs, the ouput error is expected to be sufficiently small.**

In terms of iteration index k, the weight update equations are

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j(k) x_j^{(i)} + w_0(k)\right)\right)\mathbf{x}^{(i)}$$

$$w_0(k+1) = w_0(k) + \eta \sum_{i=1}^{N}\left(y^{(i)} - \left(\sum_{j=1}^{n} w_j(k) x_j^{(i)} + w_0(k)\right)\right)$$

- **k corresponds to the epoch number, the number of times the set of N pairs is presented and cumulative error is compounded.**