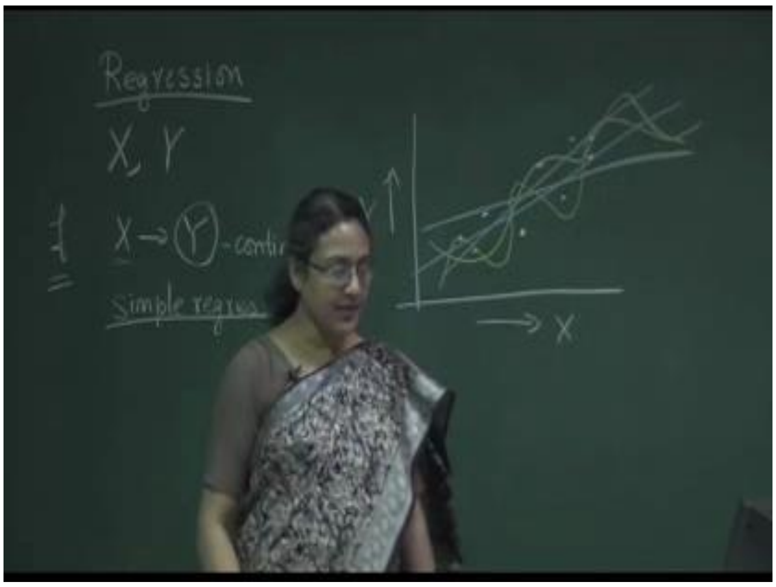
Introduction to Machine Learning

Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module – 2
Lecture – 05
Linear Regression

Good morning, welcome to the course on Introduction to Machine Learning. Today, we will start the second module of the course, which is introduction to the Linear Regression and Decision trees. These are some of the simplest machine learning algorithms; and with this, we will start looking at the machine learning algorithms. In first part of this module, we will talk about linear regression.

(Refer Slide Time: 00:50)

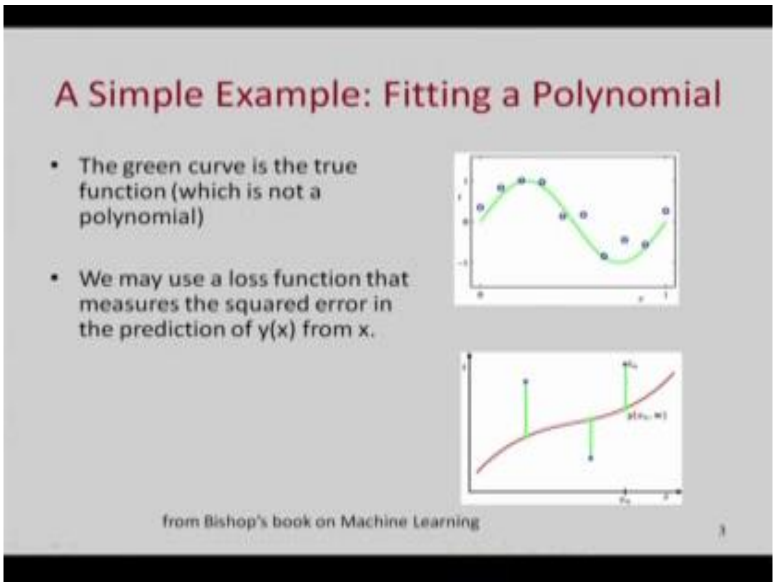


We have already explained what is regression, so regression is a supervise learning problem, where your given examples of instances whose X and Y value are given, and you have to learn a function, so that given an unknown X, you have to predict Y. So, you want a function which predicts given X predicts Y; and for regression, Y is continuous. So, this is the regression problem. And there are many modules that can be used for regression that is many types of functions can be used. The simplest type of function is a linear function. Now, X can comprise of a single future or multiple futures.

For simplicity, we will first talk about simple regression where X is a single feature. We will also discuss multiple regressions, where X comprises a number of features. So, if x is a single feature we can think of the training examples can be plotted as suppose this is the value of X, and this is the value of Y, and let us assume that both X and Y are continuous valued. So, each training example is a point in this space, for this value of X this is the value of Y; for this value of X, this is the value of Y, so like that you may have different points in the feature space.

And what you are required to do is find a function, so that given an arbitrary unknown X you can predict Y. Now the function can have different forms, for example, the function could be like this, or the function could be like this, or the function could be like this. So, there are different types of functions which are possible. In linear regression, we assume that the function is linear like this blue line here; and out of the different linear functions possible, we want to find one that optimizes certain criteria. So, if we look, but apart from linear regression, we could have other forms of regression.

(Refer Slide Time: 03:38)



A Simple Example: Fitting a Polynomial

- The green curve is the true function (which is not a polynomial)
- We may use a loss function that measures the squared error in the prediction of y(x) from x.

from Bishop's book on Machine Learning

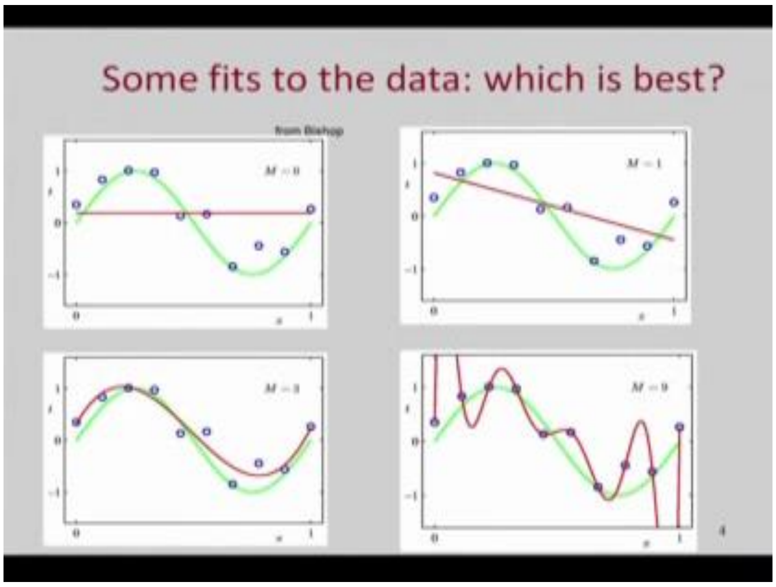
3

For example, if you look at the slide here, in this particular picture, in the first diagram, the blue circles denote the different instances in the training data. And suppose the green line is the actual function, the actual function from which the points were generated. So, in regression, you are given this blue points not the green line, and you are asked to come up with the function f. Now let us say that if you look at the diagram below, suppose these blue points are the lines, and you want to come up with a function, suppose the red line is the function that you have come up with.

Now we want to find out how could is the line with respect to the training examples. So one of the ways we can measure how good is the line is to define the error of the line. Now if you look at the blue points, so here there are three blue points, we can define the error; one way of defining the error is we find the distance of the point from this red line, and we can take the squared distance from each point to the line, and take the sum of squared errors.

The sum of squared errors is one measure of error and this is one of the popular measures of error, and we could try to find that function for which this sum of squared errors is minimized. Assuming that after we have assumed that this function comes from a particular class, you can assume that the function is linear or the function is quadratic etcetera.

(Refer Slide Time: 05:25)



Some fits to the data: which is best?

from Bishop

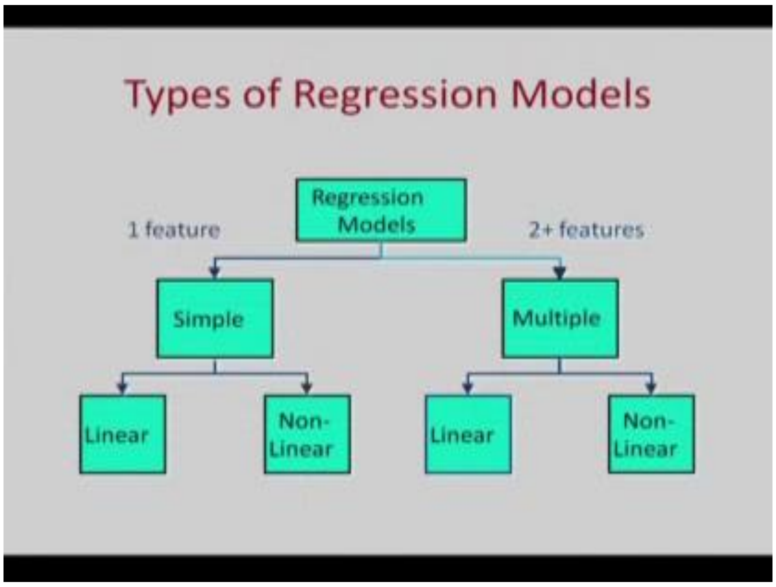
Now, let us look at this slide again. We are given these blue points, which were generated from the green curve, but the green curve is unknown to us. Given the blue points, we can learn this red line in figure 1, which is a linear function; or this linear function in figure 2, so these are two different linear functions. Sorry, this is a linear function, this is the general linear function, and this is the linear function which is parallel to the x-axis, so that is it is of the form y equal to constant.

In the first diagram, this function corresponds to y equal to constant. The second diagram, the function corresponds to the equation y equal to w x plus constant. In the third diagram, we have a cubic function, which is of the form y equal to a x cube plus b x square plus c x plus d. And in the fourth diagram, it is a ninth degree polynomial. So, this a zero degree polynomial, one degree polynomial, three degree polynomial, ninth degree polynomial.

So, if you see here the feet are not very good with the data, in the first figure. In the second figure, the feet is slightly better with respect to, if you look at the sum of squared errors this is highest in the first figure, lower in the second figure, lower in the third figure, 0 in the 4th figure. In the 4th figure, where we feet 9th degree polynomial we are able to have the function pass through all the training examples. So, the sum of squared errors on the training example is 0, but remember what we talked in the last class, what we are interested in is finding the, what is interested in is minimizing the error on future examples minimizing the error on all examples according to the distribution.

Now, you can see in this fourth diagram, even though we have fitted the points, fitted the line the red line to all the points, this function does not really correspond to the green line. So, for other points, the error may be higher. If you look at the third diagram, this function seems to have fit the points much better, and we can expect that within this range, the fit to the green line will be smaller. So, we have to keep this in mind, when we try to come up with a function.

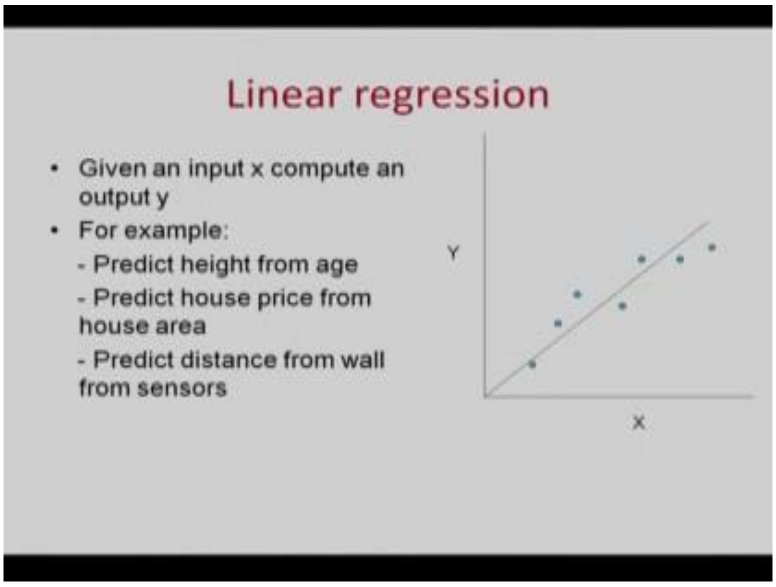
(Refer Slide Time: 08:35)



Types of Regression Models

Now regression models, as we said in regression models, we can talk about as single variable. So, x can be a single variable, then we call it simple regression; or x can be multiple variables, then we call it multiple regression. Now for each of this, the function that we defined may be a linear function or a non-linear function. Today, we will talk about linear regression where we use a linear function in order to a fit the training examples that we have got.

(Refer Slide Time: 09:16)

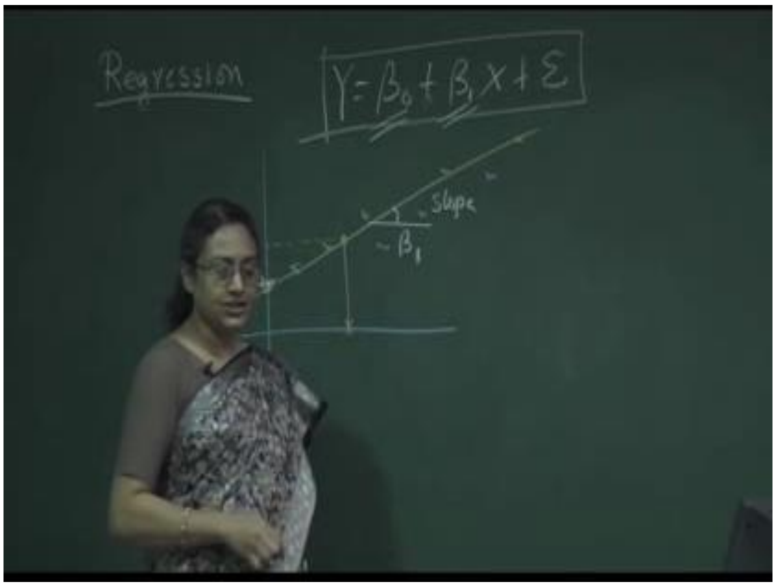


Linear regression

- Given an input x compute an output y
- For example:
  - Predict height from age
  - Predict house price from house area
  - Predict distance from wall from sensors

So in linear examples regression we have given an input x and we have to compute y.

(Refer Slide Time: 09:20)



And we have training examples which are given to us. So, we have to find a straight-line function, so that given an unknown value of x, suppose given this value of x, we have to find out what is the possible y. So, given this value of x, if you learn the yellow line, we will find out this point and from this we can find out what is the value of y as given by this function.

For example, in linear regression, we can try to fit a function from height of the person to age of a person, so that given a height, you can predict the age of the person. Or you can give the area of the house, you can predict the price of the house, or given the value of sensors, you want to predict the distance of the sensor from the wall. So you are given these blue points, and you are trying to fit as straight-line function.

Now, in this function, a linear function has certain parameters. As you know that a line can be characterized by the slope; and if we extend this line plus intercept with the x-axis. So, we can specify a line by these two parameters - the slope and the intercept with the x or the y-axis. So, if we take the equation of the line as y equal to beta 0 plus beta 1 x, we can interpret beta 0 as the point where it meets the y-axis and beta 1 as the slope of the line. So, we want to give the points, we want to find this line and finding a line means finding the parameters of the line, and let us say the parameters of the line at beta 0 plus beta 1. However, they may not exist complete fit to a straight-line.

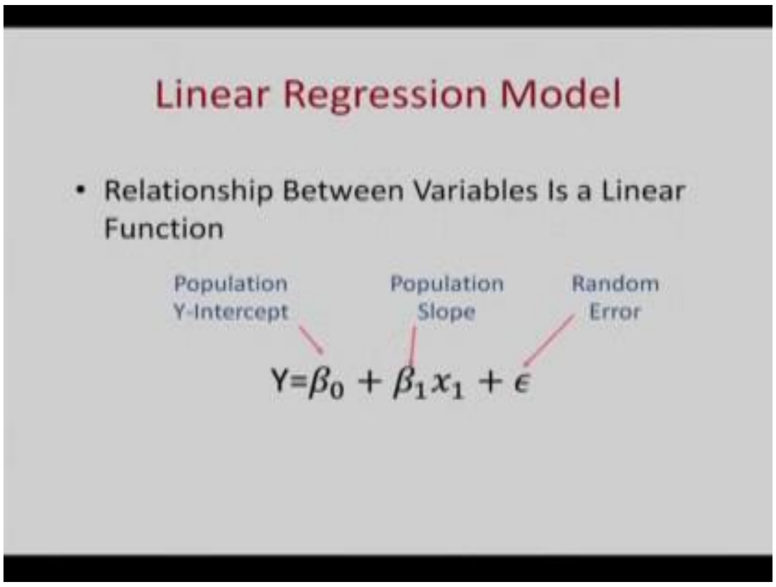
Suppose, the points are generated, so we can assume that there is some noise in the data, because we may not be able to fit the data with a straight-line. So, we can assume that there is an error, so y equal to beta 0 plus beta 1 x plus epsilon; this is the function from which the points are generated. So, we can think of there is an underline straight-line and the points are generated after accounting for some error.

For example, if you are trying to predict distance from sensor observation, there may be some error associated with the sensing device. And this error you can assume that this error has mean of zero, and some standard deviation. So, we assume that this is the underline function from which the data is generated; and given the data points, you are trying to find out beta 0 and beta 1 through which you can specify the equation of the line.

(Refer Slide Time: 13:05)

So beta 0 is the y intercept of the population line. The population line, this is the actual line, actual equation through the data is generated. So, beta 0 is the y intercept of this actual line, and we can say beta 1 is the slope of the line, or we can call that beta 1 is the population slope and epsilon is a random error.

(Refer Slide Time: 13:47)



Linear Regression Model

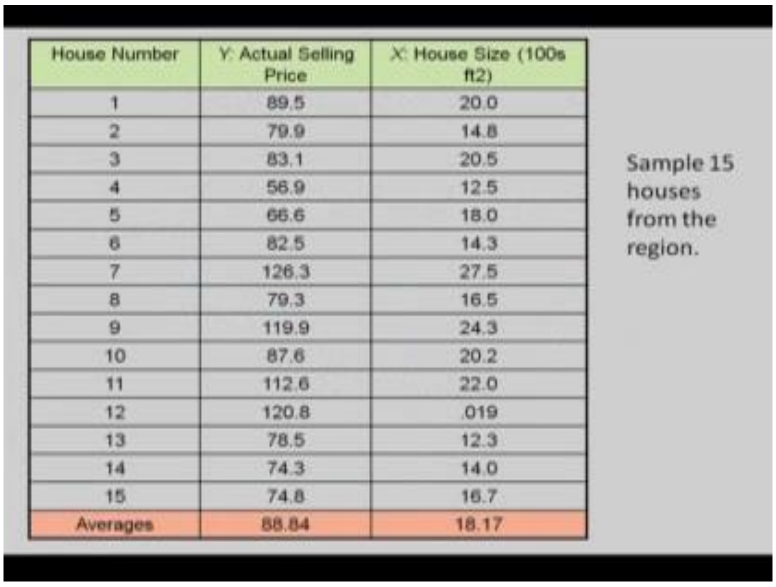
- Relationship Between Variables Is a Linear Function

Population Y-Intercept
Population Slope
Random Error

$$Y=\beta_0 + \beta_1 x_1 + \epsilon$$

Now, let us look at an example of, so this is the linear regression model, where we have this relation between the variables, which is the linear function Y equal to beta 0 plus beta 1 x plus epsilon.

(Refer Slide Time: 14:05)

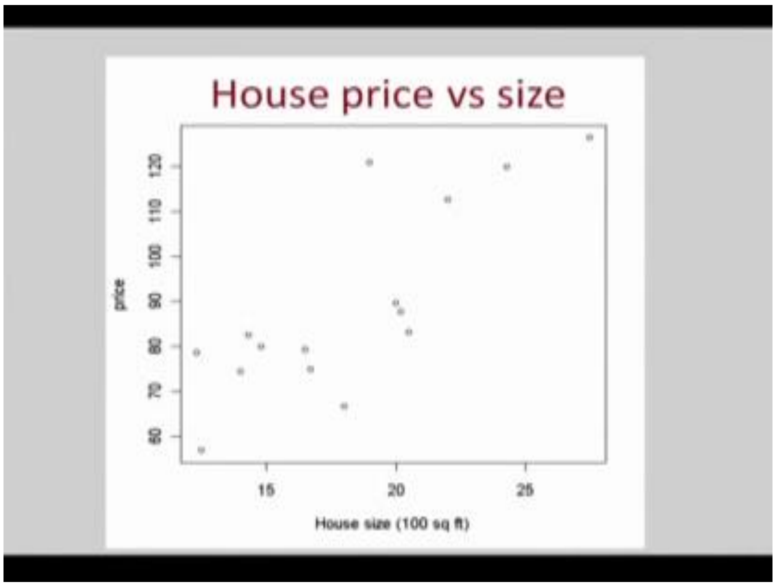


House Number	Y: Actual Selling Price	X: House Size (100s ft2)
1	89.5	20.0
2	79.9	14.8
3	83.1	20.5
4	56.9	12.5
5	66.6	18.0
6	82.5	14.3
7	126.3	27.5
8	79.3	16.5
9	119.9	24.3
10	87.6	20.2
11	112.6	22.0
12	120.8	.019
13	78.5	12.3
14	74.3	14.0
15	74.8	16.7
Averages	88.84	18.17

Sample 15 houses from the region.

And in the next slide, we look at an example of the training set. So, in this training set, we are looking at the price of a house and the size of a house. So, we are given in this particular case, we are given 15 examples; for each example, we have a house number, we have the area of the house, and we have the selling price of the house. And our objective would be given the area of the house to predict its selling price. So, what we could do is that we could plot this so x is the area of the house and y is the selling price; we can plot these functions and we can try to find the feet to this function.

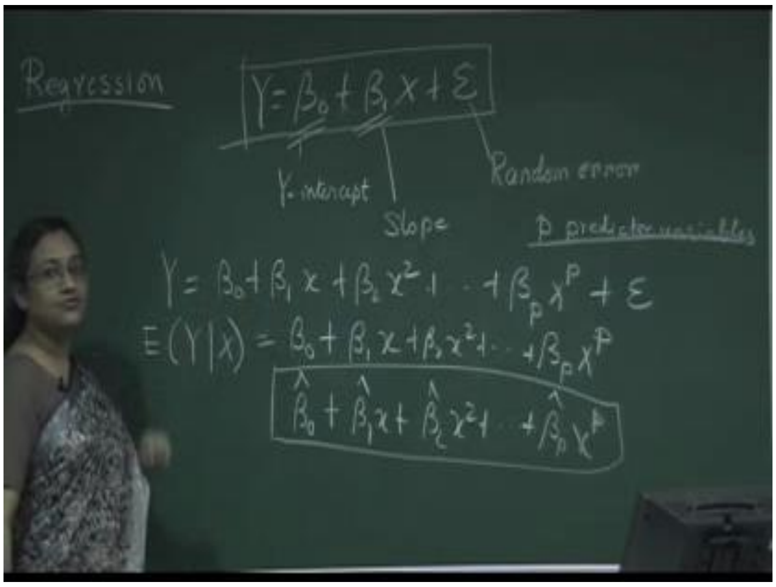
(Refer Slide Time: 14:53)



House price vs size

So, these 15 points have been plotted on this graph here, which the x-axis is the size of the house in terms of 100 square feet. So, this is the 1500 square feet, this is 2000 square feet, 2500 square feet, and this is the price of the house in lakhs. So, given these points, we want to find the equation of a line, and as we saw the equation of a line means finding the values of beta 0 and beta 1. This is for simple linear regression.

(Refer Slide Time: 15:32)

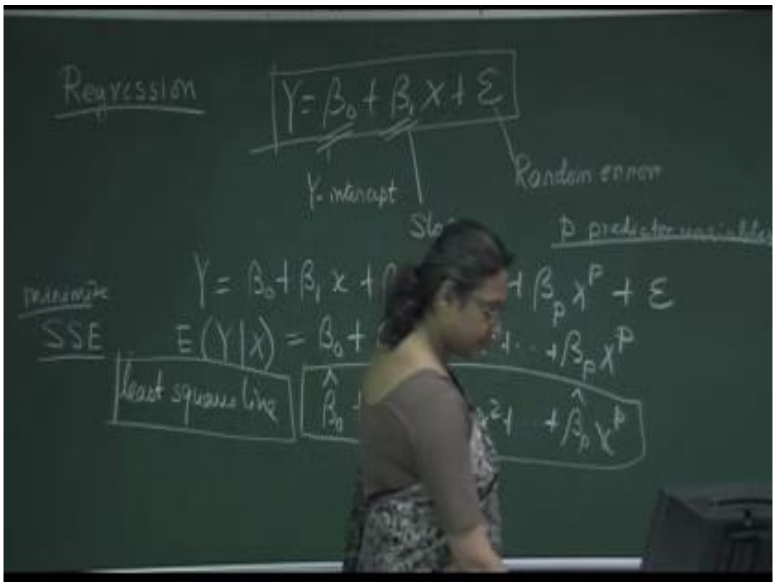

Regression
$Y = \beta_0 + \beta_1 x + \varepsilon$
Y- intercept
Slope
Random error
p predictor variables
$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \varepsilon$
$E(Y|X) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p$
$\hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \cdots + \hat{\beta}_p x^p$

For multiple linear regression, we have n variables, n independent variables, or let us say p independent variables. And we can say that the equation of the line is beta 0 plus beta 1 x plus beta 2 x square plus beta p x to the power p plus epsilon, so this is when we have p predictor variables or p independent variables. So, we have p predictor variables. So, we have to come up with the model for finding out these values of beta 0, beta 1, beta 2, beta p etcetera.

So, our model assumes that so what we are assuming is that the expected value of Y given X follows this equation. So, this equation is the equation of the population line that is the equation from which the examples are actually drawn. So, expected value of Y given X is given by the population line, or because epsilon is a random error, and we assume that the mean of epsilon is 0, we can say that expected value of Y given X is beta 0 plus beta 1 x. In case of linear, simple when we have one variable; and it will be beta 0 plus beta 1 x plus beta 2 x square plus beta p x to the power p, when we have multiple variables.

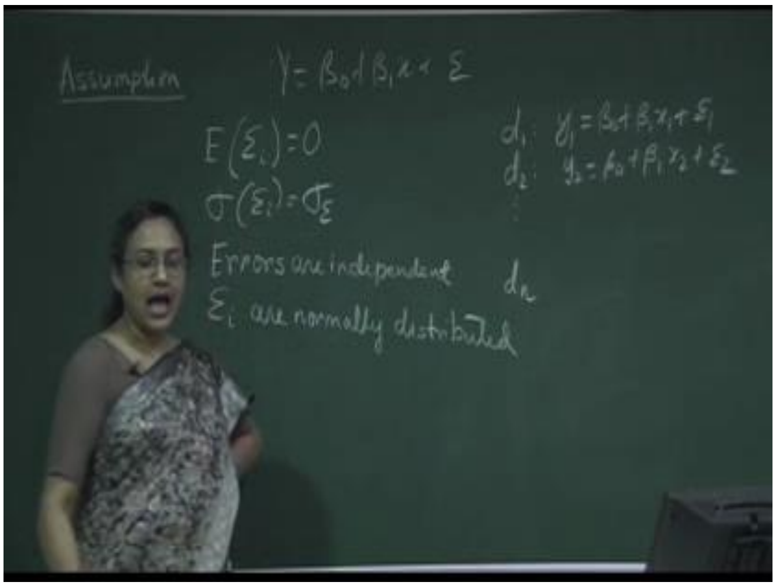
Now, given the data points, we are trying to find out the equation of the line that is an estimated value of each of this parameters. So, we are trying to come up with beta 0 hat beta 1 hat beta 2 hat beta p hat, so that the equation that we get is like this. So, this is the equation that we are trying to come up with as an estimated for the actual function actual target function, this is the actual target function. This is the function that you are trying to come up with. And we will try to optimize certain things to come up with this functions; this optimization will be with respect to the training examples that we have.

(Refer Slide Time: 18:47)


Regression
$Y = \beta_0 + \beta_1 x + \varepsilon$
Y- intercept
Slo
Random error
p predictor variables
minimize SSE
$Y = \beta_0 + \beta_1 x + \ldots + \beta_p x^p + \varepsilon$
$E(Y|X) = \beta_0 + \ldots + \beta_p x^p$
least square line
$\hat{\beta}_0 + \ldots + \hat{\beta}_p x^p$

So for example, we can try to find out those values of beta 0, beta 1, beta p, so that the sum of squared error is minimized. So, if we want to minimize the sum of squared errors, and based on that we come up with values of beta 0 hat, beta 1 hat, beta 2 hat, beta p hat this particular equation is called the least square line. So, we will see that given training points how we can come up with the least square line. So, let me just rub the board.

(Refer Slide Time: 20:08)

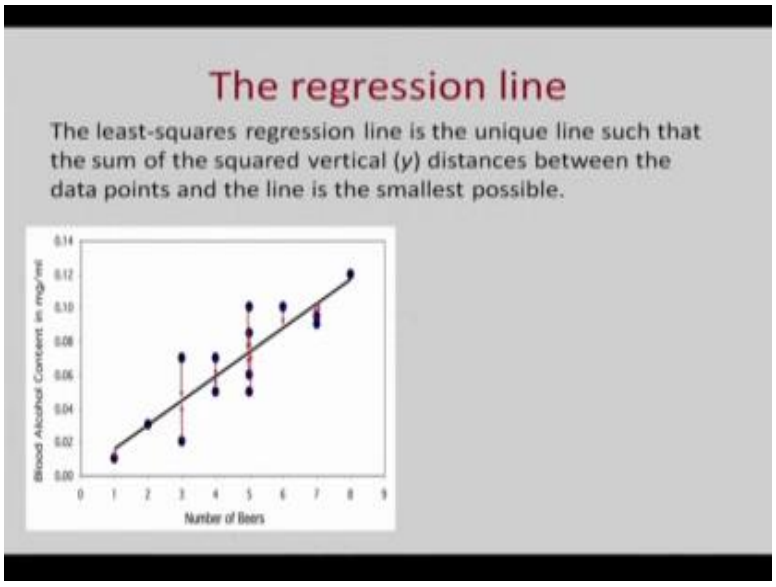

Assumption
$Y = \beta_0 + \beta_1 x + \varepsilon$
$E(\varepsilon_i) = 0$
$\sigma(\varepsilon_i) = \sigma_\varepsilon$
Errors are independent
ε_i are normally distributed
$d_1: y_1 = \beta_0 + \beta_1 x_1 + \varepsilon_1$
$d_2: y_2 = \beta_0 + \beta_1 x_2 + \varepsilon_2$
d_n

Now the data that we have may not form a perfect line. So, what we will do is that we will make some assumption about the error. So, the assumptions that we make, so let us just for simplicity take the simple linear regression Y equal to beta 0 plus beta 1 x plus epsilon. And the assumption that we make about the error is so we are getting different data points, let us say d 1, d 2, d n; and corresponding so d 1, we have y 1 equal to beta 0 plus beta 1 x 1 plus epsilon 1. d 2 comprises of y 2 x 2, where y 2 equal to beta 0 plus beta 1 x 2 plus epsilon 2. So, with respect to every example, there is a value of the error, epsilon 1, epsilon 2, epsilon 3 etcetera.

The assumption we make is that expected value of epsilon i equal to 0 that is the error that is added is has mean of 0. So, we want to find the equation so that whatever is the residual error that will have a mean of 0. And let us say the standard deviation of these errors is taken to be sigma epsilon; the sigma epsilon is unknown. So, the error is come from a distribution, whose mean is 0, and it has some standard deviation, and this standard deviation is unknown to us.

Further, we will make the assumptions that the errors are independent that is epsilon 1, epsilon 2, epsilon n, they are independent each other. And we can also assume that these errors are normally distributed they are normally distributed with mean 0, and standard deviations sigma e, so this sort of noise is called Gaussian noise or white noise.

(Refer Slide Time: 22:40)

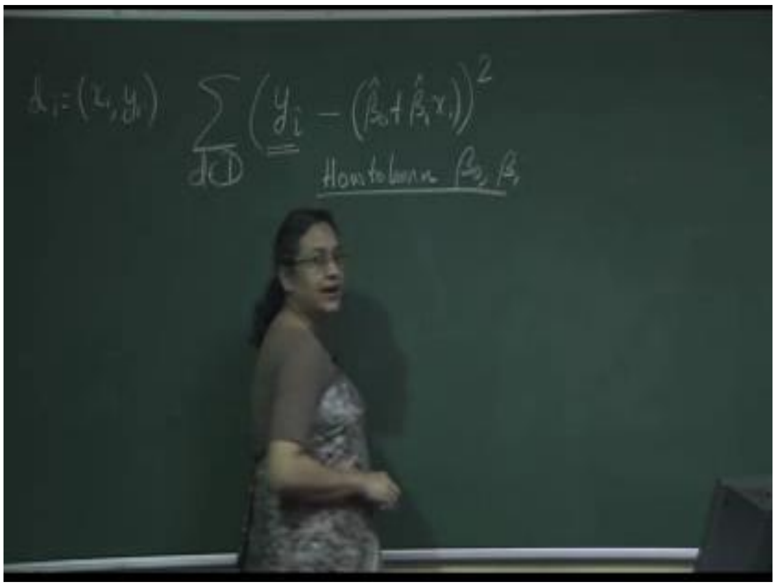


The regression line

The least-squares regression line is the unique line such that the sum of the squared vertical (y) distances between the data points and the line is the smallest possible.

Now, as we said that now given the training points, the blue are the training points in this picture, we are come up with a line, and for that line, we can find out what is the sum of squared errors with respect to the blue training points. Out of all possible lines, so the different lines are parameterized by the values of beta 0 and beta 1. For each pair of values beta 0, beta 1, we will have one line; we want to find that line for which the sum of squared errors is minimum. So, the least square that is called the least squares regression line. The least squares regression line gives the unique line such that the sum of the squared vertical distances between the data points and the line is the smallest possible.

(Refer Slide Time: 24:05)

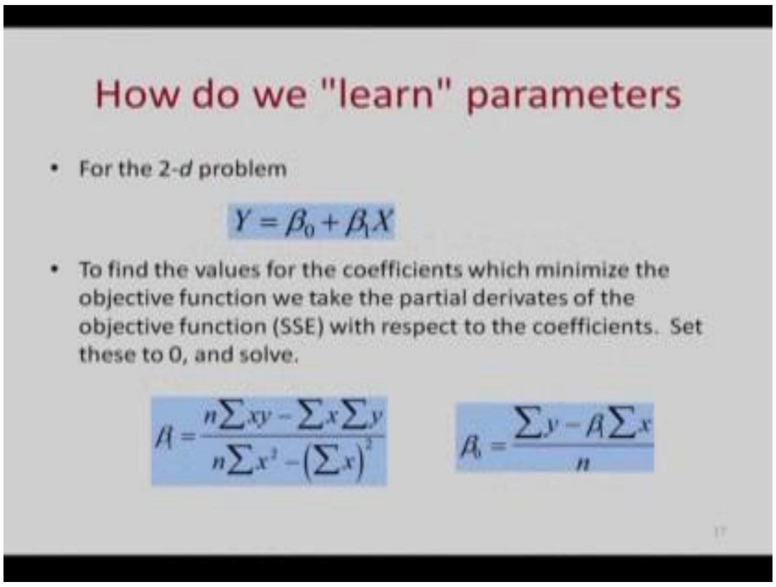

$d_i = (x_i, y_i)$
$\sum_{d \in D} (\underline{y_i} - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$
How to learn β_0, β_1

So, we will find out how to choose this line and that is the algorithm that we will develop. So, we want a line, so we have given the training points x i, y i. So, d i equal to x i, y i and we have training points like this. And we want to come up with a line so that y i minus beta 0 plus beta 1 x i. So, this is so y i is the actual value, so for a particular x i, y i should be the actual value.

And for a particular value of beta 0 and beta 1 that we have estimated, this is the predicted value. So, we want so for this particular example, the squared error is this. And over all the examples, the sum of squared errors is this; so this for all d included in the training set, this is the sum of the square errors and we want to minimize these sum of squared error given the data points x i, y i. So this is the residue that we have when we make this assumption of the values of beta 0 and beta 1. And we want to find beta 0 and beta 1, so that the sum of the squares error is minimum.

Now we have to find out how to learn the parameters. So, how to learn the parameters here the parameters are beta 0 and beta 1. And we want to find out the estimated value of these parameters. Now this problem can be solved in different ways, we can find the close forms solution given the training examples, we can find a close form solution of to get the values of beta 0, beta 1 in order to satisfy this criteria or we can come up with the iterative algorithm. So, we will look at both in this class.

(Refer Slide Time: 26:20)



How do we "learn" parameters

- For the 2-*d* problem

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

- To find the values for the coefficients which minimize the objective function we take the partial derivates of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$\beta_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2} \qquad \beta_0 = \frac{\sum y - \beta_1 \sum x}{n}$$

17

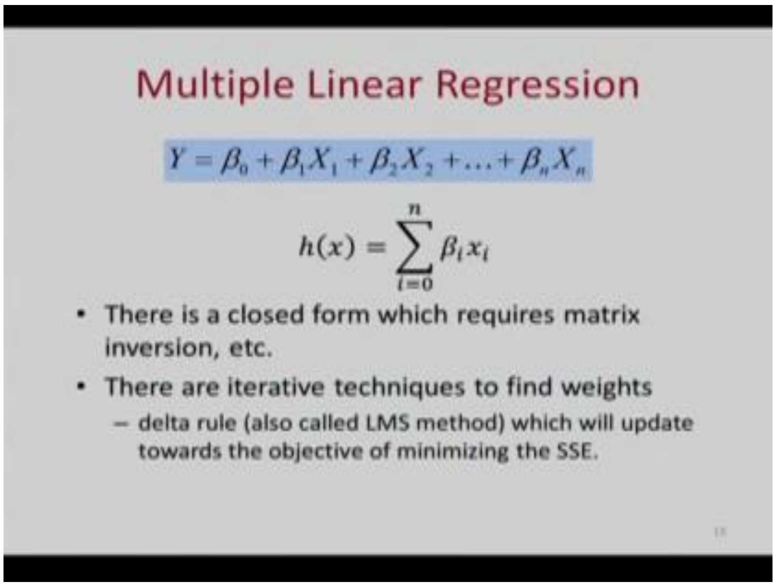
So, if we look at this two-dimensional problem, where x is a single variable. We have Y equal to beta 0 plus beta 1 X, and we want to find the values of beta 0 and beta 1 which minimizes the objective function. Then what we can do is that we can try to in order to minimize this function, we can use a standard procedure of taking partial derivative of the objective function that the one that we have written on board here.

(Refer Slide Time: 26:49)

$d_i = (x_i, y_i)$ $\sum_{d \in D} \left(\underline{y_i} - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right)^2$
How to learn β_0, β_1

If you take the partial derivative of the function with respect to the coefficients beta 0 and beta 1 and set this to 0, by solving we will get the values beta 0 and beta 1 as given in this slide. So, beta 0 is sigma y minus beta 1 sigma x divided by n; and beta 1 is n sigma x y minus sigma x sigma y etcetera. So, this derivation I am not doing in the class, but this is what you can do, and this is the close form solution that you can get.

(Refer Slide Time: 27:24)



Multiple Linear Regression

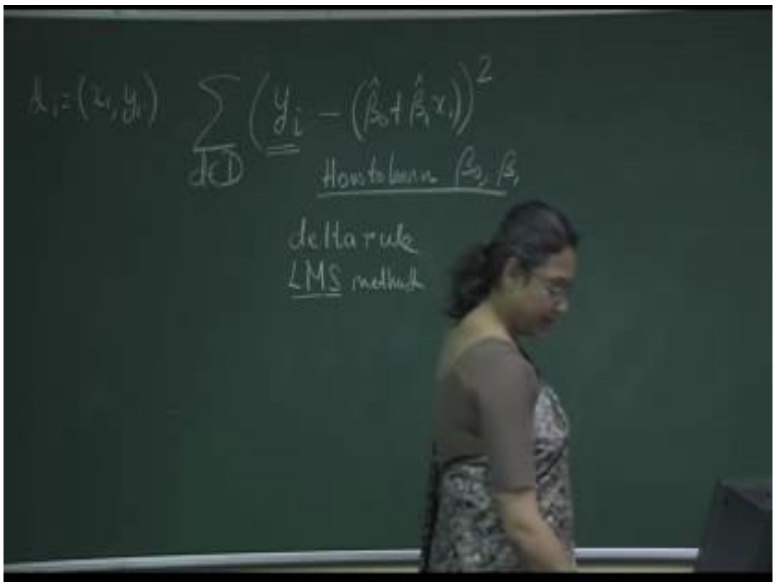
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

- There is a closed form which requires matrix inversion, etc.
- There are iterative techniques to find weights
  - delta rule (also called LMS method) which will update towards the objective of minimizing the SSE.

Now, let us come to the multiple linear regressions. In multiple linear regression, you have Y equal to beta 0 plus beta 1 x 1 plus beta n x n, or you can write h x equal to sigma beta i x i. You can here also find the close form solution; on the close form solution, we will involve metric operations in metric invention etcetera, which are involved in this. And alternative to this is to use some iterative algorithm, which will iteratively update the weight by looking at the training examples.

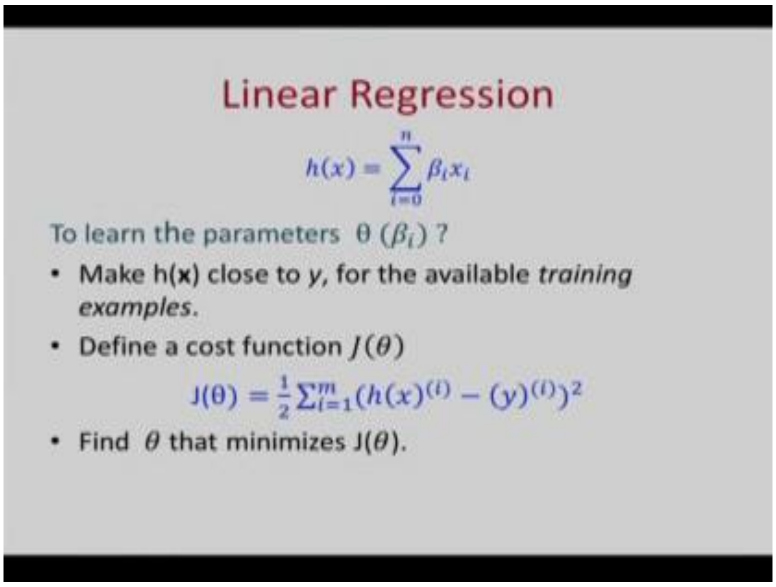
(Refer Slide Time: 28:00)



$d_i = (x_i, y_i)$
$\sum_{d \in D} (\underline{y_i} - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$
How to learn β_0, β_1
delta rule
LMS method

So, there are several iterative methods one popular well-known method is the using the delta rule which is also called the LMS method this is the same method, LMS, which stands for least minimum slope. So, this LMS method can be used. This LMS method or the delta method will update the beta 0, beta 1, etcetera weight values to minimize the sum of squared errors.

(Refer Slide Time: 28:30)



Linear Regression

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

To learn the parameters θ (β_i) ?

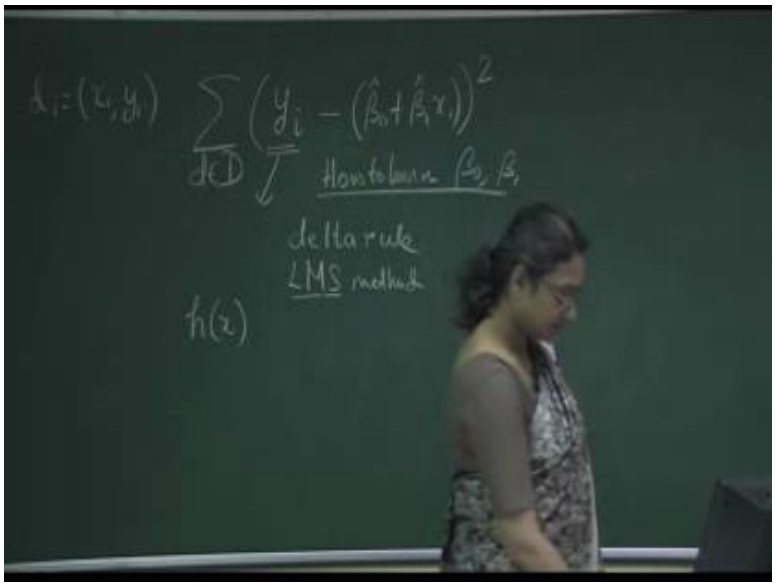
- Make h(**x**) close to *y*, for the available *training examples*.
- Define a cost function $J(\theta)$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x)^{(i)} - (y)^{(i)})^2$$

- Find θ that minimizes $J(\theta)$.

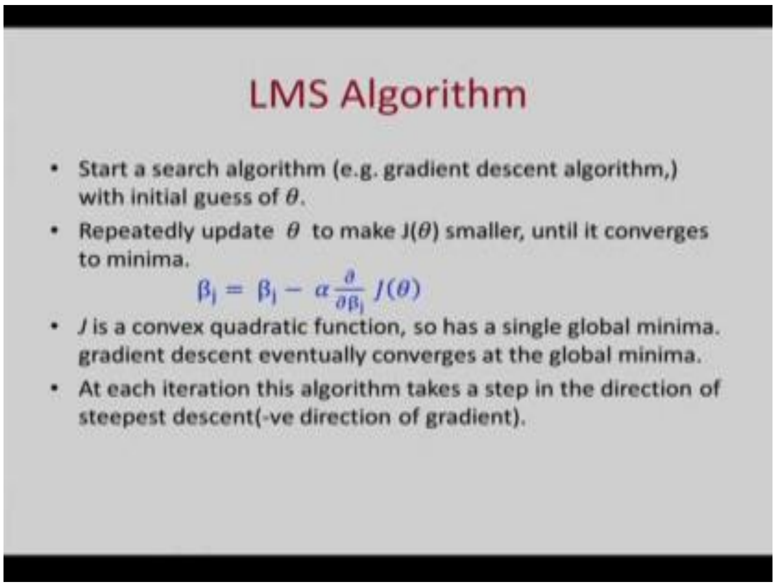
So, let us look at how it is done, so we assume that this is the form of the function, and we want to learn the parameters theta. Here, the parameters are beta 0, beta 1, beta 2 etcetera, so we want to make so we want to learn a function h x.

(Refer Slide Time: 28:46)


$d_i = (x_i, y_i)$
$\sum_{d \in D} \left(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right)^2$
How to learn β_0, β_1
delta rule
LMS method
$h(x)$

So, we want to learn about y i, so we want to learn h x which is beta 0 plus sigma beta i x i. And we had defined a cost function j theta based on minimizing the sum of square errors. So, j theta is sigma h x minus y whole square over all the training examples. So, this is the cost function that we had written. So, this is the cost function which we want to minimize and this function is a parameter of beta 0 and beta 1, and we want to find beta 0 and beta 1 to minimize this function.

(Refer Slide Time: 29:23)



LMS Algorithm

- Start a search algorithm (e.g. gradient descent algorithm,) with initial guess of θ.
- Repeatedly update θ to make $J(\theta)$ smaller, until it converges to minima.

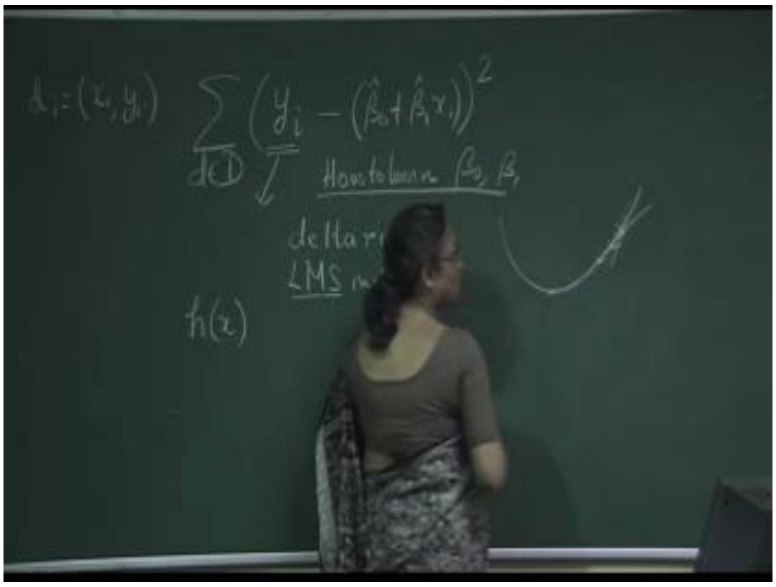
$$\beta_j = \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\theta)$$

- J is a convex quadratic function, so has a single global minima. gradient descent eventually converges at the global minima.
- At each iteration this algorithm takes a step in the direction of steepest descent(-ve direction of gradient).

Now, in the LMS algorithm, what we do is that we start with the initial value of theta that is initial value of beta 0, beta 1, then we take a training example or all training examples and update the values of beta 0, beta 1, so as to reduce the sum of squared errors. So, we can find out so what we can do is that this function is actually a convex function; and this convex function, this is a quadratic function and a quadratic function has single optima.

So, in this case, this quadratic function will have single minima. What we do is that we start at any point in this function space, and then we update the values of beta 0, beta 1, so as to reduce this values. And if we keep on reducing this, we will ultimately reach the minima of the function, so that the method that we follow is called the gradient descent method.

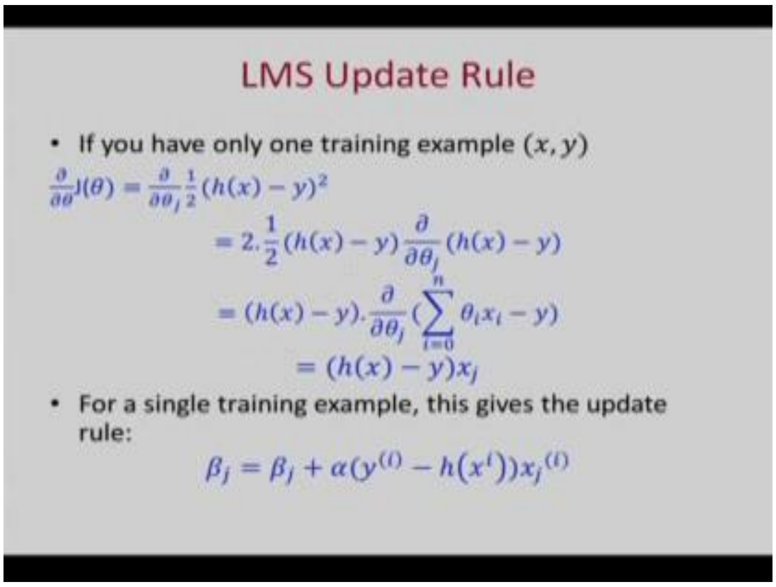
(Refer Slide Time: 30:27)


$d_i = (x_i, y_i)$
$\sum_{d \in D} \left(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right)^2$
How to learn β_0, β_1
delta r
LMS
$h(x)$

Suppose, this is a function, we start with some place here and then what we do is that we find a gradient at this point, we find the gradient of the curve at this point, and then we take a small step in the negative direction of the gradient, so this is called gradient descent. And we continue doing this, with small step, if we take small steps ultimately you will go the minima of this function.

Now so given this function, we can take a partial derivative of this function, and work out that del del beta j, J theta is the gradient of the function. And we want to take a small step in the negative direction of the gradient, and alpha here is the step size. Alpha is small and alpha determines if alpha is larger, we take larger steps; if alpha is small we take smaller steps. So, this is the initial value of beta and we make a small change to beta in the negative direction of the partial derivative of the j theta with respect to beta. And since J is a convex quadratic function, it has single global minima, and so gradient descent will eventually converges to the global minima.

(Refer Slide Time: 31:50)



LMS Update Rule

- If you have only one training example (x, y)

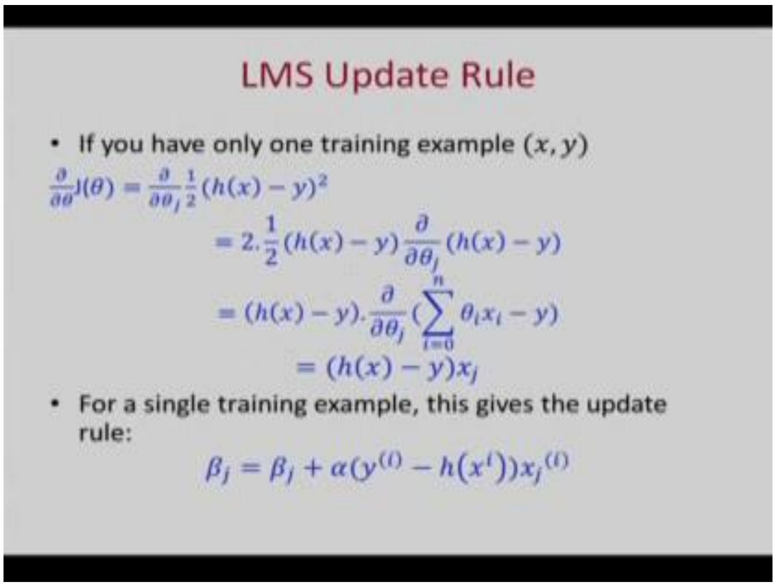
$$\frac{\partial}{\partial\theta}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{1}{2}(h(x)-y)^2$$
$$= 2.\frac{1}{2}(h(x)-y)\frac{\partial}{\partial\theta_j}(h(x)-y)$$
$$= (h(x)-y).\frac{\partial}{\partial\theta_j}(\sum_{i=0}^{n}\theta_i x_i - y)$$
$$= (h(x)-y)x_j$$

- For a single training example, this gives the update rule:

$$\beta_j = \beta_j + \alpha(y^{(i)} - h(x^i))x_j{}^{(i)}$$

So, this is the LMS update rule, if you have a single training example, you can work out del del theta J theta.

(Refer Slide Time: 31:06)



LMS Update Rule

- If you have only one training example (x, y)

$$\frac{\partial}{\partial\theta}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{1}{2}(h(x)-y)^2$$
$$= 2.\frac{1}{2}(h(x)-y)\frac{\partial}{\partial\theta_j}(h(x)-y)$$
$$= (h(x)-y).\frac{\partial}{\partial\theta_j}(\sum_{i=0}^{n}\theta_i x_i - y)$$
$$= (h(x)-y)x_j$$

- For a single training example, this gives the update rule:

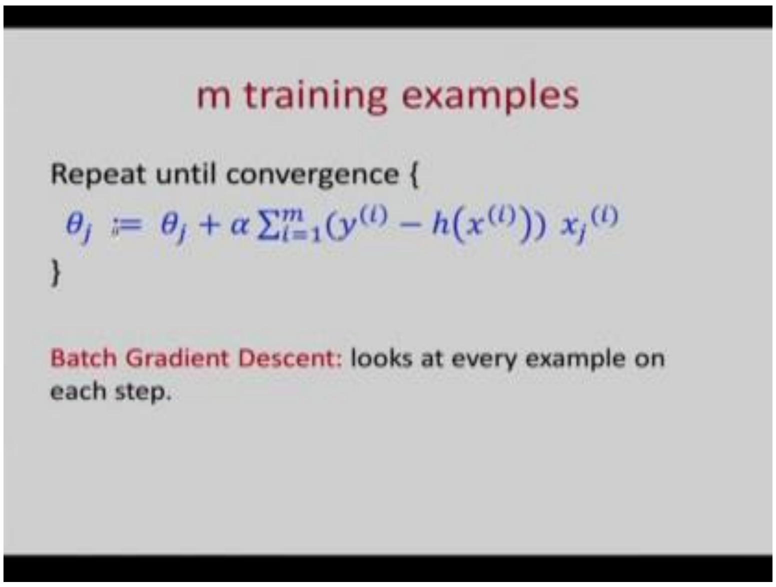
$$\beta_j = \beta_j + \alpha(y^{(i)} - h(x^i))x_j{}^{(i)}$$

So, you can work out as and we can just look at the slide here, and so del del theta. So, actually one thing I forget to tell you that in this function, we have put a half here; you know it is easy to see that whatever minimizes the rest of the function is also minimizes the half of this function, and we are just taken half for the variance it is not very important.

So, because we have taken half here then we do at take a derivative, we can work out it, it is if we take a make a change of variable h x minus y, then we can take so this is a z square, so this the derivative of its 2 into z, so we have 2 into half into z h x minus y times the partial derivative of h x minus y. And by simplifying what we get is h x minus y into x j. So, for a single training example, we get the update rule as beta j equal to earlier value of beta j plus alpha times the y minus h x i times the value of x j.

So, this is the delta rule; and this delta rule, you can easily interpret is that if y and h x are same you do not change beta, but if there are different suppose y is greater than h x then what we have to do you have to increase beta, so that h x comes closer to y. If y is smaller than h x, you have to decrease beta and that is what the LMS update rule or the delta rule is doing.

(Refer Slide Time: 34:05)



m training examples

Repeat until convergence {

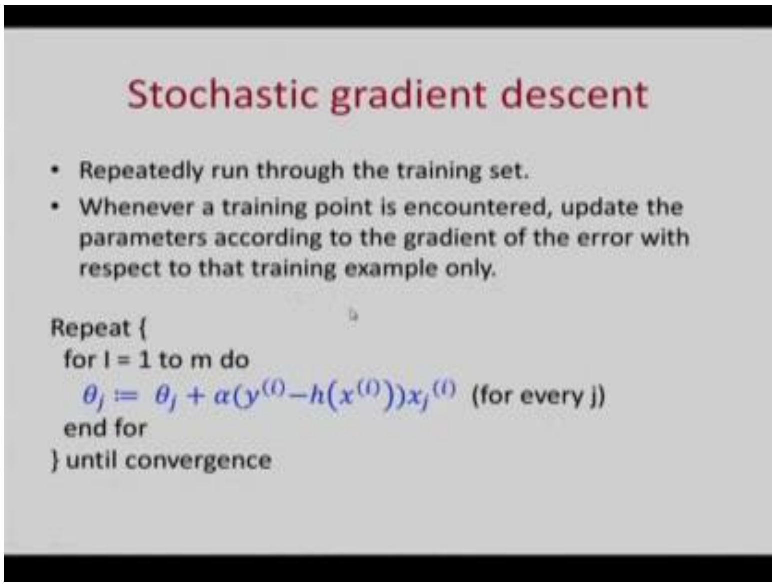
$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)}))\ x_j^{(i)}$

}

Batch Gradient Descent: looks at every example on each step.

So, you can do this for a single training example, or you can update for all the training examples at a time. So, you find out summation of this, and then you take an update, you take all the training examples and update all of them together, this is called batch gradient descent. Or you can update based on a single example which is called incremental gradient descent. So, batch gradient descent, takes the right steps in the right direction, but it is very slow because you make one step after processing all the inputs.

(Refer Slide Time: 34:51)



Stochastic gradient descent

- Repeatedly run through the training set.
- Whenever a training point is encountered, update the parameters according to the gradient of the error with respect to that training example only.

```
Repeat {
  for I = 1 to m do
    $\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x_j^{(i)}$ (for every j)
  end for
} until convergence
```

So, what is usually done is that we use stochastic gradient descent where we take examples one at a time, and make changes based on that training examples, if we do that the entire process fast and it has also been shown that stochastic gradient descent has very nice properties, so that it is does converge. And between batch gradient descent and stochastic gradient descent, we can also have mini batch gradient descent, where we take a few examples at a time, based on that; we decide the update of the parameters.

With this, we come to the end of this lecture.

Thank you.