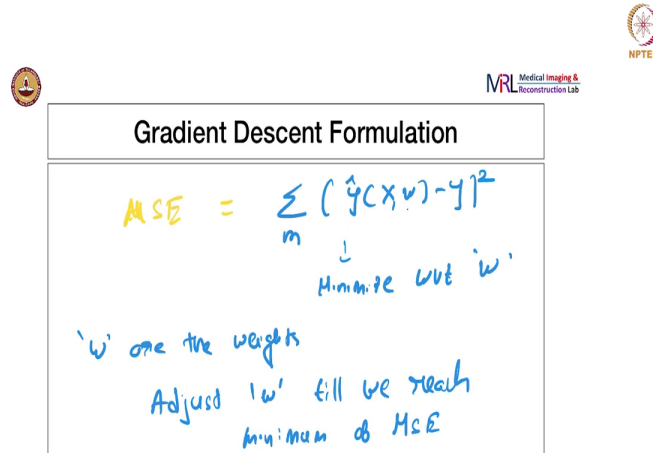


**Medical Image Analysis**  
**Professor Ganapathy Krishnamurthi**  
**Department of Engineering Design**  
**Indian Institute of Technology, Madras**  
**Lecture 39**

**Gradient Descent Formulation**

(Refer Slide Time: 00:15)



The slide features a white background with a thin black border. At the top center, the text "Gradient Descent Formulation" is written in a black sans-serif font. Below this, the Mean Squared Error (MSE) formula is presented in a handwritten style:  $MSE = \sum_m (\hat{y}(x, w) - y)^2$ . The variable  $MSE$  is highlighted in yellow, and the summation symbol  $\sum$  is in blue. A blue arrow points from the  $w$  in the formula to the handwritten text "Minimize wrt 'w'". Below the formula, the text "'w' are the weights" is written in blue. At the bottom, the instruction "Adjust 'w' till we reach minimum of MSE" is written in blue.

Gradient Descent Formulation

$$MSE = \sum_m (\hat{y}(x, w) - y)^2$$

Minimize wrt 'w'

'w' are the weights

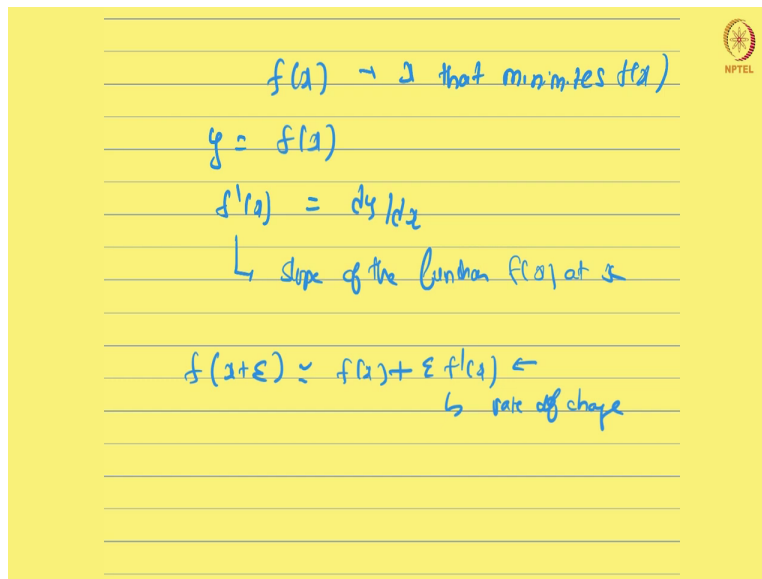
Adjust 'w' till we reach minimum of MSE

Welcome back. So, in this video, we are going to look at the gradient based optimization, specifically the steepest descent or the gradient descent optimization algorithm. So, now we saw that in the context of linear regression, we were looking at this error function or the mean squared error, which we write down and use different color, is some summation over all the data points,  $\frac{1}{m} \sum (\hat{y}(x, w) - y)^2$ , this is your ground truth or the training example  $y$ , again,  $y$  and  $\hat{y}$  are scalars, squared error.

So, this, we want to find the, minimize this function, I am going to minimize this with respect to  $w$ ,  $w$  or the weights or the parameters of the model. So, all machine learning algorithms will involve some optimization like this. This is called an optimization problem. And the idea is now we have to keep adjusting  $w$ . So, adjust  $w$ , till we reach minimum of MSE. So, there are many systematic ways of doing this and they fall under this category called optimization.

And there is, of course, there are some, there is a specific topic of optimization called context optimization, but in general we will not talk to talk about that, we will just talk about how one would go about estimating  $w$  using this gradient descent formulation, especially in the context of linear regression.

(Refer Slide Time: 02:15)


$$f(x) \rightarrow \text{a that minimizes } f(x)$$
$$y = f(x)$$
$$f'(x) = dy/dx$$

↳ slope of the function  $f(x)$  at  $x$

$$f(x+\epsilon) \approx f(x) + \epsilon f'(x) \leftarrow$$

↳ rate of change

So, if we let us say, so let us, let us, let us consider some function  $f(x)$ . So, this is a single variable,  $f(x)$  is our function, and we are trying to minimize or maximize this function, and we want to find out the  $x$ , the  $x$  that minimizes or maximizes this function. We will just say minimizes, at this point, not to get confused,  $x$  that minimizes  $f(x)$ . This is what we want to find out, this is what we want to estimate.

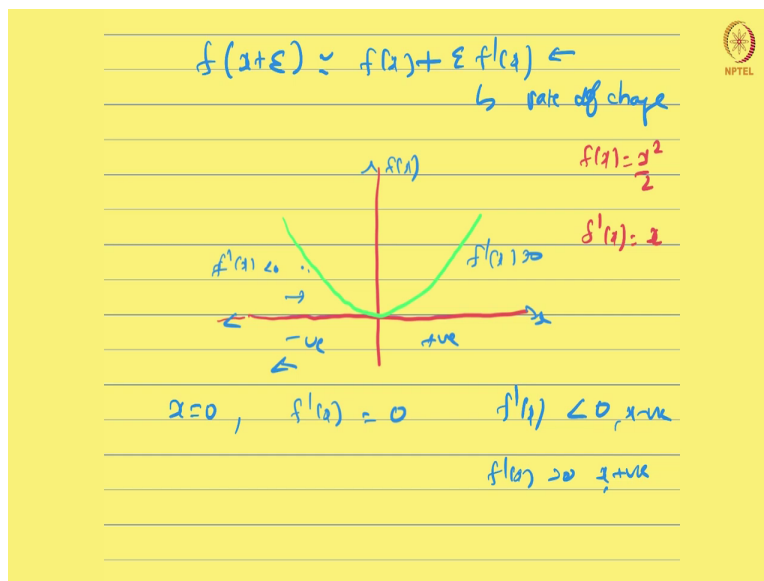
So, given this, how do you go about doing that? So, that is the question. So, for every function, like say, we are we are going to write this as  $y = f(x)$ , for every function where, in this case  $x$  and  $y$  are real numbers, there is a derivative of  $x$ . So,  $f(x)$ , let us say, smooth function, it does not discontinue this, we assume that it is derivative  $x$ . So, the derivative of this function is denoted as  $f'(x)$ , which is our  $\frac{dy}{dx}$ , these are two ways of denoting this.

Now, we know that for, in, if  $x$  is, we are looking at single variables and we know that the derivative  $\frac{dy}{dx}$  or  $f'(x)$  gives the slope, gives the slope of the function  $f(x)$  at  $x$ . That is

the slope of the function, that is, that is what we, and you are all familiar with this concept of the slope of the function. So, what is that, what is the slope of function tell you? It tells you how much  $f$  changes if you change  $x$  by a small amount. So, that is the idea.

Or in other words, the model that you are trying to, that you are formulating is  $f(x + \epsilon) \sim f(x) + \epsilon f'(x)$ , where the epsilon is a very small number. So, it is basically the rate of change with respect to  $x$ . So, locally, how fast it changes when you change  $x$ ? So, for small changes in  $\epsilon$ , now this is an approximation to this is how you calculate  $f(x + \epsilon)$ . So, which means that it is useful for minimizing the function because what this derivative tells you is that how to change  $x$  in order to change  $f(x)$ .

(Refer Slide Time: 04:48)



Now let us just look at a very in a simple function that we can draw and we can appreciate how this works. So, let us say we can draw, we can we can look at  $f(x) = \frac{x^2}{2}$ . And so,  $f'(x) = x$  so now if we draw this, one second, so just the best approximation, we can draw something of that sort. So, basically at  $x = 0$ , for this function,  $f'(x) = 0$ .

If you put  $f'(x) = x$  and it is at 0. So, when the, when the gradient or the derivative for this function is set to 0, it is, it is at a critical point or at an optimal point. In this case, it is

the global minimum. So, for negative values of  $x$ , this is negative values of  $x$  and this is positive values of  $x$ , this is  $x$  and this is  $f(x)$  of course.

For negative values of  $x$ , we can calculate  $f'(x)$  and it will turn out that  $f'(x) < 0$ , for  $x$  negative, and  $f'(x) > 0$  for  $x$  positive. So, we can just, because you just have to plug in value here. So,  $f'(x) = x$ . So, for negative values of  $x$ , our negative gradient, values, gradient value is negative, for positive value of  $x$ , gradient is positive.

So, what that tells you is that even  $f'(x) < 0$ , which means that when on this side when  $f'(x) < 0$ , on this side  $f'(x) > 0$ , means that we can move, we can, by moving to the right, we can move, by moving to the right, we can decrease the value of  $f(x)$ . And since on the hand side, when  $x$  is positive  $f'(x) > 0$ , we can decrease  $f(x)$  by moving leftward.

In a sense, we move in the direction opposite to that indicated by the gradient. So, because the gradient is, in this case, is negative. So, as we go in this, further in this direction, gradient gets more and more negative. So, we move in the opposite direction, which to, in order to minimize  $f(x)$ . So, this is the gradient descent technique.

So, basically, the way to reduce the value of the function with respect to or optimize the function with respect to its, say, variables, independent variable,  $x$ , is to move in a direction opposite, which is opposite in sign of the directive, of the derivative. So, and we can take very small steps so that we do not overshoot the optimum. So, this is the principle behind the gradient descent algorithm.

Now we are just going to see how it works for functions which are, which have multiple variables as input. So, in our case our loss function, we saw, was, was basically a function of weights, and there are multiple weights, because we have lots of features, each feature is multiplied by weight in linear regression and there is also a bias term, so the number of features plus 1, basically, and in that scenario how do we formulate this problem.

So, in the one-dimensional case, all I have to do is move in, find out the sign of the gradient and move in the direction which is opposite to the, move in sense, make small steps in  $x$  with the opposite sign of the derivative. So, that is how we reduce  $f(x)$  or

minimize  $f(x)$ . So, this is for one dimension. So, what if we have multiple variables, which have multiple inputs.

(Refer Slide Time: 09:00)

Handwritten notes on a yellow background:

Top right:  $f(w) = w_1 + w_2$

Center:  $J(w) = \frac{1}{m} \sum (\hat{y}(x, w) - y)^2$

Below the equation:  $n+1$  weights

Below that:  $J(w)$

Below that:  $\frac{dJ}{dw_i} \left[ \frac{dJ}{dw_1} \dots \frac{dJ}{dw_{n+1}} \right]$

Below that:  $\downarrow$

Below that: gradient vector

Bottom:  $J(w): \mathbb{R}^n \rightarrow \mathbb{R}, \quad \Delta w \rightarrow w + \Delta w$

So, for instance our  $J(w)$  which we had,  $J(w)$  is our loss function. So, we have  $J(w) = \frac{1}{m} \sum (\hat{y}(x, w) - y)^2$ . So, this is our  $J(w)$ , that is the function, this is our last function. So, the inputs this is like if you have  $n$  features of  $(n + 1)$  weights, multiple weights. So, in this case you have to be familiar with the concept of a partial derivative. So, for instance, if we will use  $x$  but it would be confusing so I will just move to  $w$ 's.

So, for instance, if you calculate in this case  $J(w)$ , instead of  $f(x)$ , we just use  $J(w)$ . So, a  $\frac{\delta J}{\delta w_i}$  where  $i$  is the subscript for the each of the weights, basically this tells you how  $J$  changes only with respect to this particular variable,  $w_i$ . So, then we can, we can do this for every  $w$ . So,  $\frac{\delta J}{\delta w_1}, \frac{\delta J}{\delta w_2},$  all the way to  $\frac{\delta J}{\delta w_{n+1}}$ , which you have a bias term.

So, this forms a vector, so this vector. So, this is a gradient vector that we have. So, for functions with multiple inputs, so, basically our function  $J(w): \mathbb{R}^n \rightarrow \mathbb{R}$ . So, it takes a bunch of features as input and produces a scalar output, that is what linear regression does. So, that, in that case what we want to do is take the gradient rather than the derivative, the

gradient is of course the vector of the partial derivatives with respect to each of the input variables or the variables involved in the function.

So, in this scenario, how does gradient descent look like. So, then in what direction should we go? So, for instance in one dimension, it is very simple. You find out the sign of the derivative and move in the, take steps in the direction opposite the, opposite side. That is, that is not 1D in 1D, but this is in multiple directions. So, because there now there are infinite possibilities, how, in what direction do you take the step?

When I say take the step, in how much, for instance, this  $x$  is fixed, so in this case we are trying to estimate  $w$ , so we have to change all these  $w$ 's. We have  $(n + 1)$   $w$ 's, so how do we change  $w$ 's? So, we want to estimate this  $\Delta w$  so that we can do  $\Delta w + w$  so that  $J$  is minimized. So, how do we estimate that, delta  $w$ , and what direction do we take the step.

(Refer Slide Time: 12:00)

$\Rightarrow J(w) \rightarrow$  Multiple inputs  $x$  is fixed

$$f(x+\epsilon) = f(x) + \epsilon f'(x)$$

$w$  is a vector

Directional derivative  $\rightarrow \vec{u}$   
unit vector

$$(\vec{u} \cdot \nabla_w J(w)) \rightarrow \text{Dot product}$$

$\rightarrow \vec{u}^T \nabla_w J(w) \rightarrow$  find direction in which  $J$  decreases the fastest

So, for functions like the one, like our last function or the error function, we had talked about the MSE, mean square error where  $J(w)$ , this has multiple inputs,  $w$ 's, basically. Remember that  $x$  is fixed, so even though I have written it as part of the function whenever I write  $J(w)$ ,  $x$  is fixed, the only thing that the variable is  $w$ . So, then, the idea is now, how do we know the direction of the gradient.

So, for instance in one dimensional, we just did that  $\frac{dy}{dx}$ , so when we  $\frac{dy}{dx}$ , what we did in which is we called it as frame of  $x$ , we just defined that to be this way is  $f(x + \epsilon) \sim f(x) + \epsilon f'(x)$ . This is the way to do it. So, now, if we, if you are going to do it this way, so for multiple inputs, so in this case,  $x$ , I am just switching back and forth between  $x$  and  $w$ , so please put up with me.

In this case when I use the small  $x$ , it is just some variable in one dimensions. So, now how do we take, when we have multiple inputs,  $w$  is a vector,  $w$  is a vector. Then in this case, how do we take this derivative? So, the idea is now, we want to take something called the directional derivative. How do we take directional derivative, and we want to take it in a particular direction  $u$ .

So,  $u$  is a vector, it is a unit vector, which means its magnitude is 1.  $u$  is a unit vector, its magnitude is 1, and we want to estimate the slope of the function  $J$ , here  $J$ , this function  $J$  along  $u$ . So, how would you do that? So, the mean vector notation, so the gradient of, let me erase and rewrite. So,  $(\vec{u} \cdot \vec{\nabla} J(w))$  this is a mean square error. So, we want the gradient, we saw that this was a vector, this is a vector with so many components as the size of  $w$ .

And we want to take, so we can, we will just put a vector arrow here, and we want to take a component of this vector along the unit vector  $u$ . So, you just have to do  $u$  dot, just a dot product. So, if you want a unit vector, along some unit vector if you want your estimated slope, especially in, if it has a multi-dimensional input, then this is the way to do it.

And in matrix notation this reduces to  $u^T \nabla J(w)$ . So, it is a dot product within two vectors and you can write it this way. So, now in order to minimize our  $J$ , we would like to find the direction in which  $J$  decreases the fastest. So, the idea is to find direction in which  $J$  decreases the fastest.

(Refer Slide Time: 15:50)

$$\min_{u, u^T u = 1} u^T \nabla_w J(w)$$

$$\vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos \theta$$

$$\min_{u, u^T u = 1} \|u\|_2 \|\nabla_w J(w)\|_2 \cos \theta$$

$$\uparrow$$
  
 independent of  $u$

$$\min_u \cos \theta \quad \text{So } \theta = \pi$$

$\Rightarrow u$  and  $\nabla_w J$  are pointing in opposite direction

Decrease  $J$  by stepping in the direction of   
 $-ve$  gradient

So, how do we estimate this direction? So, let us see, we just, it is not too hard. So, we want to minimize this particular expression. So, what do we want to minimize? We want to minimize,  $u^T \nabla_w J(w)$  and we want to minimize this expression with respect to  $u$ ,  $u$  is the direction, we want to estimate, of course, we know that  $u^T u = 1$  because its magnitude is 1. And the dot product, we can always write. So, if your dot product of two vectors, if you have two vectors  $\vec{a} \cdot \vec{b}$ , we know it is just  $\vec{a} \cdot \vec{b} = |a| |b| \cos \theta$  it is a scalar quantity.

So, in this case also, we can write in this particular notation, we are using, so minimize with respect to  $u$ ,  $u^T u = 1$ . We can write this as  $\|u\|_2 \|\nabla_w J(w)\|_2 \cos \theta$ . So, when is this maximum? Because we want to find the direction in which  $f$  would decrease the fastest. So, in this case  $\theta$  is the angle between  $u$  and the  $\nabla_w J(w)$ . And then of course, we know this is a unit vector, so we can say this to be 1.

And since this does not depend on  $u$ , it is independent of  $u$ , this particular expression here, I am just going to, yeah, this expression is independent of  $u$ . So, now we just have to find out, independent of  $u$ . So, now we just are left with pretty much  $\cos \theta$ . So, which means that, so we are pretty much writing something like minimum with respect to  $u$

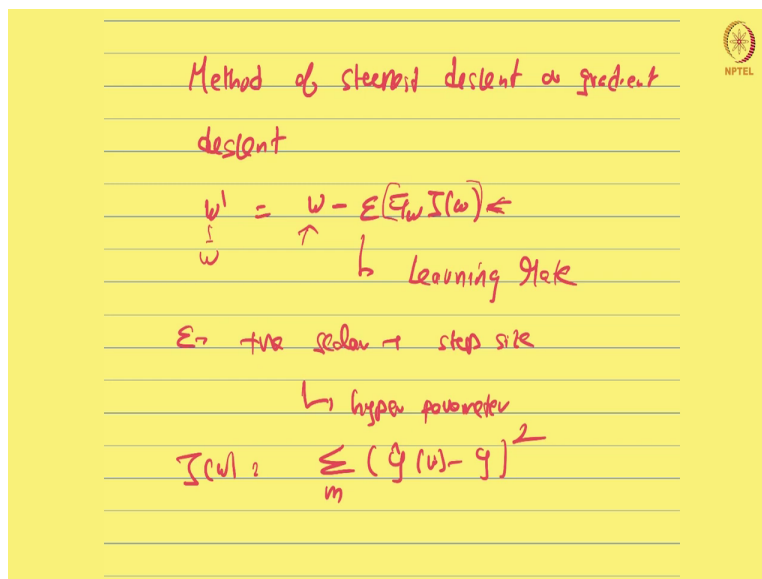


$\cos\theta$ . This is minimized when the angle between  $u$  and  $\nabla wJ(w)$  is 180 degrees. So, that is basically when  $u$  points in the direction opposite to that of the gradient.

So, so in which case, so which is, which is meaning that the gradient points in the direction in which the function is increasing and the negative of the gradient points directly in a direction where the function is decreasing. So, then we can, so which means, so it is minimizing, it is when  $\cos\theta = -1$  this side, and  $\theta$  is 180 degrees, which tells you that  $u$  should be the direction opposite to the gradient, which means that we can decrease  $J$  by moving in the direction of the negative gradient. This is known as the method of steepest descent.

So, however, this is, which implies  $\theta$  or which make  $\cos\theta = -1$  which means that that  $u$  and gradient of  $J$  with respect to  $w$  are pointing in opposite directions. And then, this gradient of  $J$ , and then for the way to decrease the  $J$  with respect to  $w$  because, by stepping in a direction of the negative gradient, so decrease  $J$  by stepping in the direction of negative gradient.

(Refer Slide Time: 19:52)



Method of steepest descent or gradient descent

$$w' = w - \epsilon \left( \frac{dJ}{dw} \right)$$

$w$        $w'$        $\epsilon$  learning rate

$\epsilon$  → the value of step size

↳ hyper parameter

$$J(w) = \frac{1}{2} \sum (y(i) - \hat{y}(i))^2$$

So, this is called the method of steepest descent or gradient descent, and the formula is the following. So,  $w' = w - \epsilon \nabla wJ(w)$  where this  $\epsilon$  is called the learning rate. Again,

this notation might be different, some people might use just, we have already used lambda, so I am refraining from using  $\lambda$ . So, we can use  $\beta$  or some other alphabet.

So, this basically a positive scalar. Epsilon is a positive scalar, this one is a positive scalar, some number, which basically constrains your step size. It constrains how big a step we will take. But this is also a hyper parameter. So, we talked about hyper parameters. This is a hyper parameter. And it is one of those things that you would optimize in most machine learning algorithms.

So, we calculate this gradient numerically, and in many cases and then we keep moving by, and then of course, remember that say  $w' = w - \epsilon \nabla w J(w)$ . The second step, this would be the new  $w$  and then you would continue doing this iteratively, and then you stop once, yes you do not change or  $w$  does not change, appreciated, but our  $w$  does not change in a in a very, by a large amount. So, you have to fix that threshold also.

So, therefore, many tricks associated with it, we will not go into detail because it is a very well established literature, a lot of variations of this algorithm is available in many software packages and you just have to know how to use them, but this is the underlying principle. Generally, you calculate the gradient of the function and then you would point, you, it would it, would point uphill basically in the way function is increasing, whereas if you want to decrease the function, you just move in the opposite direction.

Now, this might look slightly difficult to comprehend why this happens, but if you, one way of understanding this is using the 1D example. So, if just plot some  $x^2$ , calculate the value of, or some function in 1D, and then you calculate the values of the derivative and see in which direction and if you see, look at its sign and based on that you can understand which way, in which way you have to move in order to arrive at the opt or the minimum of that function

Now, this is all assuming that of course, I am just making an assumption that this is all has a global minima. There also functions with multiple and optimum. Some of them could be a maxima. Even at the maximum gradient, goes to 0, but it is kind of unstable, but still you might go, get stuck there, you might, you might have to climb that, and then

there are, there, there are these, in the case of neural networks, etc. they might have multiple minima.

So, usually, you have to start off with some  $w$ , with some guess of  $w$ , and from there you have to refine your  $w$ . And this again, is, you can treat this like a hyper parameter, you can start off from different values of  $w$ . But then there are all these algorithms which try to compensate for that is like you start off with any arbitrary  $w$ , will get you there to a local optimum.

So, in many machine learning algorithms, especially neural networks,  $w$  generally converges to a global optimum. It is only for so-called convex functions, the one, there is only one true optima and that happens to be the global optimum or the global minimum. So, this is a gradient descent. Now, what we want to do for real-time learning is we want to do something called stochastic gradient descent.

We will just briefly look at what stochastic gradient descent does, but then before we go any further, this is  $\epsilon$ , when I, when we do this, this, this step, this gradient of  $\nabla w J(w)$ , if you think about this, this is evaluated over all the training data. So, this is something, because your last function  $J(w)$ , I am going to rewrite this, sorry, let me, let me just rewrite this. So,  $J(w) = \sum (\hat{y}(w) - y)^2$ .

So, now what we are trying to, so what we are, what you will get here is some average gradient over all these data points. That is what we will get when you do this expression. In fact, in this case, you can actually directly, directly calculate the gradient, if you, if you did, if you take the derivative with respect to  $w$ , you will be able to, be able to calculate it. We are not going to get into that here, but the idea is for gradient descent, you will consider all the data at one time.

And then, we are actually using the average gradient here, kind of, to take the step. So, if you have a very large data set, you have to use all those points to calculate this one step every time. So, that is why, we, even this form of gradient descent is probably not suitable when you have real-time data or when you have very large data sets. So, we will look at a variation of this, called stochastic gradient descent, and I think a mini batch

gradient descent. Now, and sometimes I think this method is also referred to as batch gradient descent. So, we will be looking at this mini batch gradient descent and stochastic gradient descent.

(Refer Slide Time: 26:00)

$$\nabla_w J(w) = \frac{1}{m} \sum_{i=1}^m \nabla_w J(x^{(i)}, y^{(i)}, w)$$

Stochastic Gradient Descent

mini batch size  $m'$

$$\widetilde{\nabla_w J(w)} = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_w J(x^{(i)}, y^{(i)}, w)$$

$$w \leftarrow w - \epsilon \widetilde{\nabla_w J(w)}$$

So, we were talking about the previous slide, we said if you look at the gradient  $\nabla_w J(w) = \frac{1}{m} \sum \nabla_w J(x^i, y^i, w)$  it is actually the average, you just compute this as an average, 1 to m. Now, this m becomes very large, you have to do this for every data point.

So, as a training set becomes big, in many cases, this training set can run into hundreds of millions or maybe billions of examples, then this becomes computationally much harder to do. So, the stochastic gradient descent is the preferred, is the go-to algorithm here. Instead of doing this for the entire data set, what is done is to actually consider a so-called mini batch of examples.

So, this is like, the understanding this is the following, so here I said, now you are kind of doing an average or ideally, I mean you should call this the expectation, but we will just work with averages. So, you are calculating empirically, you are approximating the expectation with an average for this particular case. Now, we have very large number of samples.

So, what this so-called mini batch stochastic gradient does is, so for every calculating step size, we do not have to estimate this average on the entire data, but rather a sample of the data. So, we do an average of the sample and use that to update the weights of your linear model. So, the mini batch size, which is  $m'$  as we call it, which is a subset of  $m$ , is chosen to be a generally small number.

We can be anywhere, we want usually powers of 2 sometimes, 1, 32, 64, 100, something like that but you will have tens of thousands of examples, but you might just use a mini batch of 50 to 100 to, for every step. So, that, this is, you, you choose these 100 randomly. And you can use that, and it, and in fact, the cost to compute, do a step given that you fix that  $m'$ , so the mini batch size is  $m'$ , which is a very small number.

It remains the same regardless of how large your data set is, so then, but you can continue this step, take steps at the same cost, no matter how large your data set. So, then the gradient estimate, which we call, which is basically  $\nabla w \tilde{J}(w) = \frac{1}{m'} \nabla w \sum J(x^i, y^i, w)$  this is your last function,  $J$  of, in, for individual data points.

So, once you have this gradient estimate, mini batch gradient estimate then you can use your usual  $w$  goes to, and I am going to just, I know, I do not know, put a giant tilde here to, this, and  $w \rightarrow w - \epsilon \nabla w \tilde{J}(w)$  your mini batch gradient estimate. So, that is your stochastic gradient descent.

So, surprisingly works very well for a wide variety of problems, optimization problems. In fact, a lot of the deep learning techniques that we see in the next, over the next couple of weeks, they all use this algorithm for estimating the weights of their models or the parameters of the model.

And not only for linear regression but for a wide variety of algorithms. And it is a, it is a workhorse of modern machine learning techniques. So, again, we are not going to look at, consider how, whether it will converge, what are they, what are the samples, theoretical analysis, etc. Just to understand the, just the basis on which we, we use these algorithms, this is good enough.