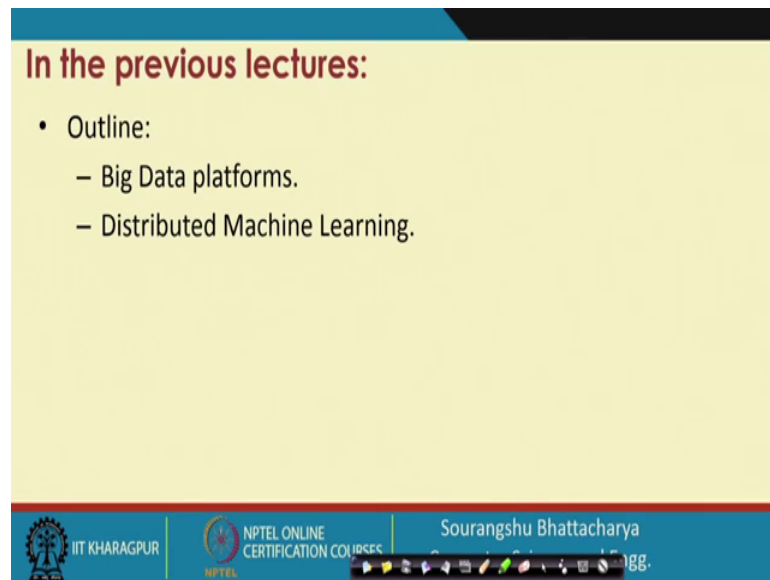


Scalable Data Science
Prof. Sourangshu Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 21a
Stochastic Optimization

Hello, everyone. Welcome to the twenty first lecture of NPTEL course on Scalable Data Science. I am Professor Sourangshu Bhattacharya from Computer Science and Engineering Department at IIT, Kharagpur. So, today we will be discussing about Stochastic Optimization.

(Refer Slide Time: 00:36)



In the previous lectures:

- Outline:
 - Big Data platforms.
 - Distributed Machine Learning.

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and the name Sourangshu Bhattacharya. A navigation bar with various icons is visible at the bottom of the slide.

So, in the previous lectures, we have discussed about the big data platforms and we have done programming on both hadoop and spark and we have seen the internal and we have also discussed about the scalable and distributed machine learning.

(Refer Slide Time: 00:54)

In this Lecture:

- Outline:
 - Stochastic gradient descent (SGD)
 - SGD Convergence
 - Minibatch and Distributed SGD
 - Practical considerations
 - Advancements from SGD.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Sourangshu Bhattacharya

In this lecture, we will discuss about the stochastic gradient descent, then we will look at the convergence of stochastic gradient descent, then we shall see the mini batch and distributed version so, stochastic gradient descents, then we shall see some practical considerations. And finally, we will see some advance recent advancements or some improvements that have come over the last 5 years or so, on SGD which make them all the more interesting towards machine learning and deep learning, ok.

(Refer Slide Time: 01:39)

Much of ML is optimization

Linear Classification

$$\arg \min_w \sum_{i=1}^n \|w\|^2 + C \sum_{i=1}^n \xi_i$$

s.t. $1 - y_i x_i^T w \leq \xi_i$
 $\xi_i \geq 0$

Maximum Likelihood *observed*

$$\arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(x_i)$$

IID

K-Means

$$\arg \min_{\mu_1, \mu_2, \dots, \mu_k} J(\mu) = \sum_{j=1}^k \sum_{i \in C_j} \|x_i - \mu_j\|^2$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, just so, we have already seen this so, much of machine learning is optimization. So, for example, linear classification; so, the first problem we are looking at here is linear classification with support vector machines and basically what happens is after you go through the model specifications and loss formulation, you end up with this problem where you want to calculate the parameters w which are your optimization variables given the data set. So, given the datasets x_i and y_i you have to calculate the w and the x_i which are the slag variables for finally, performing the classification using support vector machines.

Another very important class of methods is the maximum likelihood parameter estimation. So, given any probabilistic model p_θ which is basically defining a probability distribution over the random variables x_i ; x_i are the random variables which are the observed quantity. So, these are the observed quantities. So, for example, this could be the input in case of regression model or these could be the input in case of any general problem for example, something like collaborative filtering model and so on and so forth or some classification model and so on and so forth.

So, for any random observed variable, you can define a probability distribution p_θ over these observed variables, where this probability distribution is a function of some parameters θ . So, for example, in case of logistic regression the parameters happen to be the weight vectors. And, then you calculate what is called the log likelihood of the parameters which happen to be or rather the log likelihood which is a function of the parameters, but also a function of the input data set which is of size n and then with respect to this input data set you want to calculate the whole log likelihood and then which is the sum over the individual instances of the data set and this you can derive under the assumption that your examples are IID that is independent and identically distributed.

And, once you have this form of log likelihood your job is to basically find the parameter θ which maximizes this log likelihood. So, this is called the maximum likelihood estimate and this also turns out typically to be an optimization problem and then there are unsupervised learning problems like K-Means clustering where also your objective function is can be written as a optimization problem.

(Refer Slide Time: 05:14)

Stochastic optimization

- Goal of machine learning :
 - Minimize expected loss

$$\min_h L(h) = \mathbb{E}[\text{loss}(h(x), y)]$$

given samples $(x_i, y_i) \quad i = 1, 2, \dots, m$

- This is Stochastic Optimization
 - Assume loss function is convex

Handwritten notes: $(x, y) \sim P$ and $\sum_{i=1}^m \text{loss}(h(x_i), y_i)$

So, what is stochastic optimization? So, one way to think of machine learning problems is that you want to minimize the expected loss of a particular set of a particular set of random variables. So, let us say your input random variables are x and y which is the typical case in case of supervised learning, where y is the label and x is the input features and you want to minimize a loss function of a hypothesis h , which is the function that you want to learn basically h is the function that you want to learn and the loss function provides a kind of penalty for not exactly predicting the random variable y and then you want to minimize this loss function with respect to h , ok. Given the samples now you are given the samples x_i and y_i from a particular distribution.

Now, here I have written this term expectation. So, what we are trying to say here is that in general, we want our x and y are coming from some probability distribution P . So, they are coming from some probability distribution P and you want to minimize this loss function expected value of this loss function when x and y are drawn from this probability distribution. But, what you end up doing is you end up getting these samples x_i and y_i from this probability distribution and then you minimize the empirical loss function which is basically the summation over i is equal to 1 to m and then loss of h of x_i comma y_i , ok. So, this is the empirical loss function.

Now, the question comes that can we address the original problem of minimizing the expected loss. This also ties in with the generalization error and so on and so forth and

so, for the time being we can assume that the loss function is convex for simplicity. So, so we will see how we can go about doing this ok.

(Refer Slide Time: 08:14)

Batch (sub)gradient descent for ML

- Process all examples together in each step

$$w^{(k+1)} \leftarrow w^{(k)} - \eta \left(\frac{1}{n} \sum_{i=1}^n \frac{\partial L(w, x_i, y_i)}{\partial w} \right)$$

where L is the regularized loss function

- Entire training set examined at each step
- Very slow when n is very large

Handwritten notes on the slide:

$$\frac{\partial}{\partial w} \sum_{i=1}^n \text{loss}(h(x_i), y_i)$$

$$\leftrightarrow \sum_{i=1}^n \left(\frac{\partial L(h(x_i), y_i)}{\partial w} \right)$$

The slide also features the IIT Kharagpur logo, NPTEL Online Certification Courses branding, and a small video inset of a presenter.

So, the first thing that we do is what is called the batch gradient descent, right. So, in case of batch gradient descent what we do is we calculate the derivative of the loss function which is same as the. So, the derivative of the total loss function, so, derivative of the so, if we were to let us say differentiate with respect to w the sum over i is equal to 1 to n and loss of h of x_i comma y_i . This is same as just taking sum over i is equal to 1 to n and then derivative of loss of h of x_i comma y_i . So, these two are equivalent, because derivative is you can because the derivative function is the sum derivative or sum of functions is sum of derivative of functions.

So, using that property we can write it like this ok. So, this is our batch gradient descent or sub gradient descent as the case may be when you want to minimize the regularized loss function, ok. So, here notice that every update. So, this k determines an update for the parameters. Notice that every update requires you to calculate this loss over the entire training set. So, over all n you have to calculate the loss, ok. So, this is very slow when n is very large.

(Refer Slide Time: 10:27)

Stochastic (sub)gradient descent

- “Optimize” one example at a time
- Choose examples randomly (or reorder and choose in order)
 - Learning representative of example distribution

for $i = 1$ to n :

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

Handwritten notes: $E[\text{loss}(h(\cdot), y)]$ and $\hookrightarrow \frac{1}{\text{sample}}$

Slide footer: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES

Video inset: A small video window showing a person speaking.

So, what is the alternative? We have already looked at it briefly. So, the alternative is to use the stochastic sub gradient descent. Now, what is stochastic sub gradient descent? It is kind of you can think of it as you optimize one example at a time. So, you remember that our problem of minimizing the log likelihood is the same as you know the optimizing the expected loss or our problem of learning is same as minimizing the expected loss.

Now, it is a question of how many samples you want to take from this expected loss. So, one simple alternative is you take one sample at a time and minimize, ok. So, that is what the stochastic sub gradient descent is. So, you choose the samples in random order. So, you do not have any particular order ok, you choose the samples in random order and then your update is just using x_i and y_i . So, you are using one example to calculate loss only on one example and then you are using that to update the parameter w , ok. So, this is your stochastic gradient descent.

(Refer Slide Time: 12:04)

Stochastic (sub)gradient descent

for $i = 1$ to n :

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

- Equivalent to online learning (the weight vector w changes with every example)
- Convergence guaranteed for convex functions (to local minimum)

Perceptron

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, this is so, this is the update. So, for 1 to n so, you can iterate and now your iterations can be also over all the examples. So, your iterations are now divided into epochs. So, one epoch is when you have iterated over all the examples and then you can do multiple epochs also in random order, ok. So, basically we will come to how to practically actually implement SGD and how things work practically, but another way of thinking about the same process is that you are doing some kind of online learning, ok.

So, what is online learning? Online learning means that whenever you get one single example for training you immediately update the parameters, ok. So, for example, in so, for example, you are trying to train a spam filter. So, instead of you know looking at a whole data set of spam emails and non spam emails, you can just look at one spam email and update the parameters and then the next spam mail comes again you look at that and you update the parameter to once more. So, this is the main idea behind online learning that instead of looking at a batch of examples you will look at one example at a time, ok.

Now, stochastic gradient descent allows you to in some sense to perform this online learning also. So, this is in some sense online learning algorithm. So, the great thing about, so, there has been many many online learning algorithms for example, there has been the very famous online learning algorithm called perceptron which perceptron which is a very famous online learning algorithm for classification, ok. So, there have been many many online learning algorithms, but not all of them we are guaranteed to

always converge ok. So, for example, it is well known that perceptron diverges in case of in case of the training data set which is not linearly separable, ok.

So, so, in this case stochastic gradient descent the advantage of this is it is guaranteed to converge for all convex loss function. So, we will go through the convergence proof in a moment, but just note that it is not clear that convergence will always. So, in what sense the convergence will happen is a little bit you have to be careful, ok.

(Refer Slide Time: 15:14)

Stochastic gradient descent

- Given dataset $D = \{(x_1, y_1), \dots, (x_m, y_m)\} \rightarrow$
- Loss function: $L(\theta, D) = \frac{1}{N} \sum_{i=1}^N l(\theta; x_i, y_i) \rightarrow$
- For linear models: $l(\theta; x_i, y_i) = l(y_i, \theta^T \phi(x_i)) \rightarrow$
- Assumption D is drawn IID from some distribution \mathcal{P} .
- Problem:

$$\min_{\theta} L(\theta, D)$$

The slide includes a footer with the IIT Kharagpur logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset of a presenter is visible in the bottom right corner.

So, basically this is your stochastic gradient descent that. So, we will now go into the convergence proof of stochastic gradient descent before going into other things. So, so, here for simplicity what I will describe is somewhat simpler proof. So, somewhat different version of SGD for which we will provide the proof of convergence because it is simple to provide the proof, ok. So, as usual you have a data set of m points and you have the total loss function which is a sum of L loss functions and you have the parameter θ and you have the input x_i and y_i .

Now, we can assume that for linear models your predictor is nothing, but some $\theta^T \phi(x_i)$ even though this is not exactly necessary, for the proof to go through, but we can assume this, ok. And, now we assume that this data set D is drawn IID from some distribution \mathcal{P} . So, this is the first assumption we have to hold we have to make, ok. So, we our proof will work under this assumption and of course, our problem is to

minimize the total loss function which is a function of both the parameter and the data set and we have to minimize it with respect to the parameter theta.

(Refer Slide Time: 16:56)

Stochastic gradient descent

- Input: $D \rightarrow$
- Output: $\bar{\theta} \rightarrow$

Algorithm:

- Initialize $\theta^0 \rightarrow$
- For $t = 1, \dots, T \rightarrow$

$$\theta^{t+1} = \theta^t - \eta_t \nabla_{\theta} l(y_t, \theta^T \phi(x_t))$$

$\bar{\theta} = \frac{\sum_{t=1}^T \eta_t \theta^t}{\sum_{t=1}^T \eta_t}$

Handwritten annotations: $\Theta^T \rightarrow (\theta^1, \theta^2, \dots, \theta^T)$ and $\rightarrow (\eta_1, \eta_2, \dots, \eta_T)$ with $\sum \eta_t$ below.

And, this is the algorithm. So, the algorithm is you have the input theta a input D and you have to output theta bar, ok. Now, this is your output learnt parameter you initialize the parameter theta to theta 0 and then for you run this algorithm for capital T many time steps and for capital T many time steps you do this update which we have discussed also earlier that your theta t plus one becomes theta t minus this eta t and then gradient of the loss function with respect to theta at that point in time, ok. So, whatever the parameters of theta are at that point in time the gradient with respect to that, ok.

And, then this is one interesting change how usually the stochastic gradient descent is implemented is that instead of returning. So, normally you would just return theta capital T which is the final theta ok, but now instead of returning that theta capital T we are returning the weighted average of the step size weighted theta T and then the divided by the summation of the step size. So, you can think about this as you are you have this theta 1, theta 2 and then theta capital T you have this many parameters which you have seen throughout the stochastic gradient descent algorithm, and now you are you have also seen the learning rates which are of course, positive which is eta 1, eta 2 and like this eta t eta capital T.

And, now you divide them all by you know sum over eta t, ok. So, you are basically making a probability distribution out of this learning rate and then you are taking the weighted average with respect to this probability distribution as the final output. So, instead of putting all your weight on the final distribution you are just taking a weighted average over all the distributions, ok. So, you can clearly see that if this theta T provides. So, what we will show ok. So, let me go to the next one ok.

(Refer Slide Time: 19:49)

SGD convergence

- Expected loss: $s(\theta) = E_{\mathcal{P}}[l(y, \theta^T \phi(x))]$
- Optimal Expected loss: $s^* = s(\theta^*) = \min_{\theta} s(\theta)$
- Convergence: $E_{\bar{\theta}}[s(\bar{\theta})] - s^* \leq \frac{R^2 + L^2 \sum_{t=1}^T \eta_t^2}{2 \sum_{t=1}^T \eta_t}$
- Where: $R = \|\theta^0 - \theta^*\|$
- $L = \max \nabla l(y, \theta^T \phi(x))$

Handwritten annotations: (n, T) , Best loss, Reported loss, $T \rightarrow \infty$, $\sum_{t=1}^T \eta_t \rightarrow \infty$, 0 finite, $\sum_{t=1}^T \eta_t^2 < \infty$.

So, now $s(\theta)$ is the expected loss which we defined in the first class. So, with respect to this probability distribution \mathcal{P} from which this x and y are drawn we are taking this weighted average of the whole loss function with respect to this, and then this the minimum with respect to θ we are calling that s^* . So, this is the kind of the best loss, ok. So, the best expected loss that you can get and this is the reported loss in some sense it is a reported loss, right.

So, at the end of running the algorithm the loss that you would get expected value of that. Now, why is there an expectation? Expectation is because depending on the data set your θ may change, right. So, because you are taking a different permutation of the data set and so on, your $\bar{\theta}$ may change and then the question is that if you have if you have if you take expectation with respect to all possible such ways of you know taking $\bar{\theta}$ and everything else and then calculate the loss which is this expected loss over the data points then this difference. So, so this is going to be the minimum by definition

because this is where it is minimum. So, the difference from the minimum; so, this is going to be somewhat bigger than this the amount by which this is going to be bigger is bounded by this quantity, ok.

Now, what is this quantity? So, the first quantity is that this R which is the difference between the initial θ_0 and θ^* , ok. So, initial starting point and the so, our initial parameter and the optimal set of parameters for this problem, ok. So, this is a finite number R , ok, this is not very far. So, if you are starting somewhere so, if you are this is your space and you are starting somewhere here and this is your θ^* this is your θ^* ok, then and this is your θ_0 . So, whatever this difference is and the length of that is this R and L is another number which is the maximum of gradient of loss, ok. So, the maximum loss you can incur the gradient of that. Again, this is also we are assuming that this is bounded because you cannot give infinite loss to some number. So, both of the both R and L are bounded number.

If this is the case now what it is saying is that as your iterations increase as you go from t is equal to 1 to capital T this number is bounded by this. So, $R^2 + L^2$ times this number which is summation of η^2 divided by this number which is summation of η . So, now, when will this when so. So, suppose now I may capital T tend to infinity and under this limiting case I make this following assumption that my summation over t is equal to 1 to capital T η also tends to infinity. So, I choose my learning rate parameter η such that if I take infinite many time steps the summation over the learning rate will also go to infinity, but summation over t is equal to 1 to capital T η^2 is finite. So, this is strictly less than infinity. So, this is some finite number ok. So, this is some finite number ok.


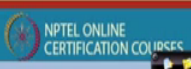

If this is the case then as you can see this term will go to 0, because the denominator will go to 0 and the numerator is some finite number, ok. So, it may be a very large finite number, but it is a finite number, but the denominator as you increase capital T will go to 0. So, the ratio will the denominator will go to infinity. So, the denominator will go to infinity because this thing will go to infinity, but your numerator remains finite, so, the whole thing goes to 0. Hence this difference goes to 0; hence we can say that this expected loss of this quantity is very close to the actual optimal loss that we can get. So, in this sense in the same expected sense the stochastic gradient descent algorithm converges ok.

(Refer Slide Time: 25:58)

SGD convergence proof

- Define $r_t = \|\theta^t - \theta^*\|$ and $g_t = \nabla_{\theta} l(y_t, \theta^T \phi(x_t))$
- $r_{t+1}^2 = r_t^2 + \eta_t^2 \|g_t\|^2 - 2\eta_t (\theta^t - \theta^*)^T g_t$
- Taking expectation w.r.t $\mathcal{P}, \bar{\theta}$ and using $s^* - s(\theta^t) \geq g_t^T (\theta^* - \theta^t)$, we get:
 $E_{\bar{\theta}}[r_{t+1}^2 - r_t^2] \leq \eta_t^2 L^2 + 2\eta_t (s^* - E_{\bar{\theta}}[s(\theta^t)])$
- Taking sum over $t = 1, \dots, T$ and using
 $E_{\bar{\theta}}[r_{T+1}^2 - r_0^2] \leq L^2 \sum_{t=0}^{T-1} \eta_t^2 + 2 \sum_{t=0}^{T-1} \eta_t (s^* - E_{\bar{\theta}}[s(\theta^t)])$

Handwritten notes on the slide:
 $r_{t+1}^2 = \|\theta^{t+1} - \theta^*\|^2$
 $= \|\theta^t - \eta_t g_t - \theta^*\|^2$
 $= \|\theta^t - \theta^*\|^2 + \eta_t^2 \|g_t\|^2 - 2\eta_t g_t^T (\theta^t - \theta^*)$

Now, how will we show this. So, first we define these two quantities first is this r_t which is this residual of θ^t minus θ^* . So, this is in some sense how far away is your t -th iterate of the parameter which is θ^t from the optimal parameter, and the second quantity is of course, what is the value of your gradient, g_t . So, this is given by g_t . Now, if we plug in so, we want to calculate what is r_t square, ok. So, r_t square is nothing, but or rather we want to calculate r_{t+1} square, ok. So, this is nothing, but θ^{t+1} minus θ^* whole square.

Now, we want to write the update the gradient descent update, ok. So, what is the gradient descent update? It is θ^t minus η times g_t or η times g_t rather and then minus θ^* whole square. Now, if we take so, we want to club these two terms together so, and then if we club these two terms together you see that it becomes r_t . So, this is something. So, so let me write it out. So, we have this equals whole square plus $\eta_t^2 \|g_t\|^2$ minus twice $\eta_t g_t^T (\theta^t - \theta^*)$ minus θ^* , ok. Now, this is nothing, but what we have here ok. So, this is just r_t square. So, this term is just r_t square, ok.

(Refer Slide Time: 29:01)

SGD convergence proof

- Define $r_t = \|\theta^t - \theta^*\|$ and $g_t = \nabla_{\theta} l(y_t, \theta^T \phi(x_t))$
- $r_{t+1}^2 = r_t^2 + \eta_t^2 \|g_t\|^2 - 2\eta_t (\theta^t - \theta^*)^T g_t$
- Taking expectation w.r.t $\mathcal{P}, \bar{\theta}$ and using $s^* - s(\theta^t) \geq g_t^T (\theta^* - \theta^t)$, we get:

$$E_{\bar{\theta}}[r_{t+1}^2 - r_t^2] \leq \eta_t^2 L^2 + 2\eta_t (s^* - E_{\bar{\theta}}[s(\theta^t)])$$
- Taking sum over $t = 1, \dots, T$ and using

$$E_{\bar{\theta}}[r_{T+1}^2 - r_0^2] \leq L^2 \sum_{t=0}^{T-1} \eta_t^2 + 2 \sum_{t=0}^{T-1} \eta_t (s^* - E_{\bar{\theta}}[s(\theta^t)])$$

Handwritten notes on the right side of the slide:

- $g_t^T (\theta^t - \theta^*)$
- $S \rightarrow \text{convex}$
- $\leq s^* - s(\theta^t)$
- $\|g_t\|^2 \leq L^2$

Logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom of the slide.

So, now we want to bound this term expectation of r_{t+1}^2 minus r_t^2 and we assume. So, we as so, we assert that this is nothing, but $\eta_t^2 L^2$ plus this term. Now, how do we get this? So, note that from here now we have to provide a kind of lower bound or rather a kind of upper bound on these $g_t^T (\theta^* - \theta^t)$, ok. So, we want to provide this upper bound over this or in other word we want to provide an upper bound over this $\theta^t - \theta^*$. So, we want to provide an upper bound over $\theta^t - \theta^*$, ok. And, this is nothing, but because of the because of the convexity of the loss function S because S is a convex loss function we have already seen that this means that this is going to be greater than or equal to or rather this is going to be less than or equal to $S^* - S^t$.

So, you see that. So, this is what is written here and if you apply this inequality and the other inequality that you apply is this norm of g_t square. So, norm of g_t square is less than or equal to L^2 . So, if you apply this then you get that you get that you get this following inequality, ok. Now, we want to take summation over t is equal to 1 to T . So, what will happen? So, on the left hand side you see that in the in the previous so, so in the. So, if we take summation over all t we will we will be left with only the first or only the first and the last residuals because r_t will appear in two terms r_{T-1} will appear 2 in 2 terms and so on and so forth and each of them will get cancelled, ok.

In this case you will just have a summation over this eta t squared, and similarly in this case you will also have a summation over t is equal to 0 to T minus 1 this term to is outside because 2 is not dependent on t.

(Refer Slide Time: 31:22)

SGD convergence proof

- Using convexity of s :

$$\left(\sum_{t=0}^{T-1} \eta_t \right) E_{\bar{\theta}} [s(\bar{\theta})] \leq E_{\bar{\theta}} \left[\sum_{t=0}^{T-1} \eta_t s(\theta^t) \right]$$
- Substituting in the expression from previous slide:

$$E_{\bar{\theta}} [r_{t+1}^2 - r_0^2] \leq L^2 \sum_{t=0}^{T-1} \eta_t^2 + 2 \sum_{t=0}^{T-1} \eta_t (s^* - E_{\bar{\theta}} [s(\bar{\theta})])$$
- Rearranging the terms proves the result.

Handwritten notes on the slide:

- $s^* = E_{\bar{\theta}} [s(\bar{\theta})]$
- $\geq -L^2 \sum_{t=1}^T \eta_t^2 + s^*$
- $2 \sum \eta_t$

Now, one more time we use the convexity of s sorry. So, one more time we use the convexity of s . You see in the previous in the here there is a summation over t is equal to 0 to T minus 1, ok. What we want is we want this summation to go inside because this θ^t is the only thing that is dependent on t or rather yeah. So, so we want this summation to go inside, ok. So, how does this summation go inside? So, so, we do two things first you note that summation over t is equal to 0 to T minus 1 η_t expected value of $\bar{\theta}$ $s(\bar{\theta})$ is less than summation over expected value over $\bar{\theta}$ t is equal to 0 to T minus 1 $\eta_t s(\theta^t)$, ok.

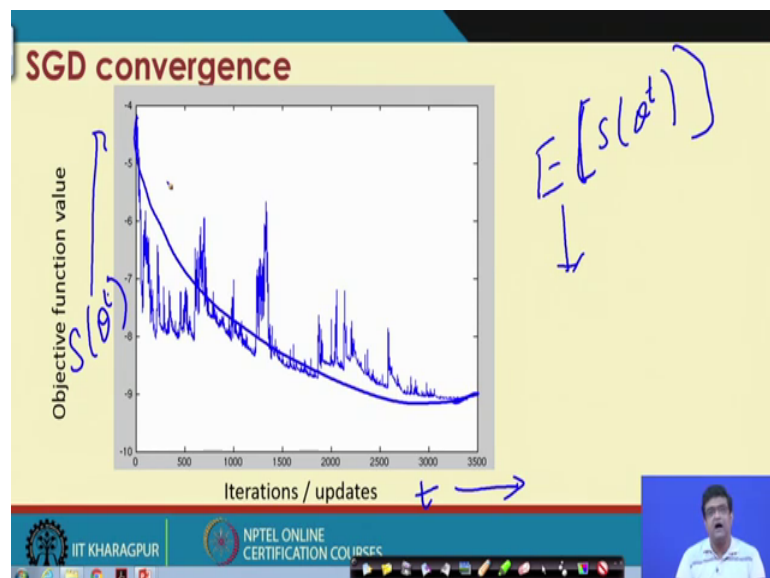
So, this comes because of inequality which is sometimes also called Jensen's inequality which says roughly that if you take a convex combination of a convex combination of some numbers, and then the expectation of that. So, expectation is a linear operator. So, if you take convex combination of some numbers, or rather if you take a function of convex combination of some num some functions, and if you take the convex combination of the function of numbers, ok. The convex combination of function of numbers is always going to be lower than the function of convex combination in other words.

If you have f or let me put it this way that if you have something like $\alpha f(a) + (1 - \alpha) f(b)$ is always going to be less than your f at the convex combination. So, in so, basically this is going to be just one times $f(a) + f(b)$ by 2 sorry. So, this is not correct. So, so just so, the convex combination of a function is always less than the rather the yeah, convex combination of a function is always less than the function of the convex combination. So, with this we are able.

So, so substituting this in the expression in the previous slide and then we have the following inequality. So, if you just rearrange. So, what you have to show is that your $s^* - \mathbb{E}[s(t)]$ is always rather this is always going to be greater than or equal to; so, this minus this is actually what we want to look, but this is always going to be greater than or equal to your L square summation t is equal to 1 to capital T ηt^2 rather minus of this plus this expectation of s . So, so this is nothing, but. So, we are calling this number as just s . So, you have here the s squared and by summation of ηt and 2 times, ok.

So, so this proves the result that we had seen in the previous lecture. So, I just let me just go back and. So, this proves this particular result that we had discussed, ok.

(Refer Slide Time: 38:17)



So, so, I will just give you a glimpse of how the convergence of SGD looks like. So, here on the x-axis we are plotting the iterations or updates of the objective function value or the count of the iteration and update. So, as your algorithm is running. So, in other words

we are plotting here the index t . So, this is the t axis and this is your s of θ_t , ok. So, you can see that it is not always minimizing, but if you take the expectation of your expectation of s θ_t something like this, ok. So, on an average this is going down, ok. So, in the average sense or in the expectation sense iterates are growing down.

So, if you do many of these iterations and take an average then you will see a trend something like this, yeah. So, you will see at you will see a trend something like this. So, we will look at some of the practical aspects of how to make a SGD work in practical scenario in the next class, but hopefully in this lecture; we have given you an intuition and also a formal proof as to why for a stochastic optimization problem the stochastic gradient descent algorithm works.

Thank you.