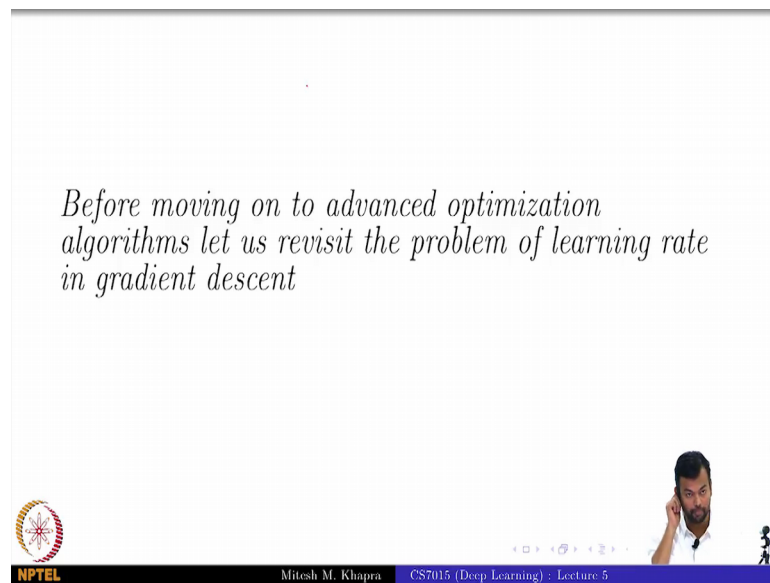


**Deep Learning**  
**Prof. Mitesh M. Khapra**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Module – 5.7**  
**Lecture - 05**  
**Tips for Adjusting Learning Rate and Momentum**

So Tips for Adjusting the Learning Rate and the Momentum.

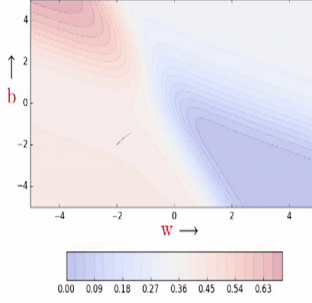
(Refer Slide Time: 00:17)





So, before moving on to these slightly advanced optimization algorithms, we will revisit the problem of learning rate in gradient descent ok.

(Refer Slide Time: 00:24)

- One could argue that we could have solved the problem of navigating gentle slopes by setting the learning rate high (i.e., blow up the small gradient by multiplying it with a large  $\eta$ )



$\eta$





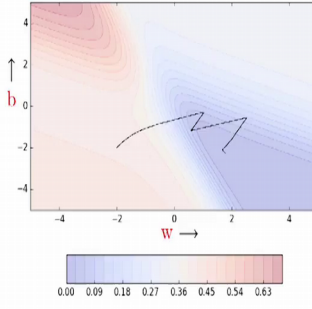
NPTEL Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 5

So, one could have argued that we could have solved this problem of this slow movement on the gentle slope by increasing the learning rate. Remember that we have this eta and we deliberately chose to be conservative, that we will take a small value for the eta, but what if I just blow up the eta; I could just take a very large eta, what would happen? It will overshoot right.

So, what will happen is, I will see what happens when I take eta equal to 10 ok. So, so I will see what happens, when I take eta equal to 10.

(Refer Slide Time: 00:57)

- One could argue that we could have solved the problem of navigating gentle slopes by setting the learning rate high (i.e., blow up the small gradient by multiplying it with a large  $\eta$ )
- Let us see what happens if we set the learning rate to 10
- On the regions which have a steep slope, the already large gradient blows up further
- It would be good to have a learning rate which could adjust to the gradient ... we will see a few such algorithms soon



NPTEL Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 5

So, this is step 1, step 2, step 3. It's moving very fast on the regions where the slope is gentle, but it also moves very fast, much faster on the regions where the slope was already steep right

So, when the gradient was actually high, you ended up blowing it further by multiplying it with the eta which is 10 right. So, it is again going to have this effect that you will move much faster in the steeper regions and algorithm you will see these oscillations, because you will overshoot your objective. Does that make sense right? So, it is not that you can always choose a high eta and get away with it

So, what do you actually want, what is your wish list regulate theta. You want an adaptive eta right that it somehow figures out that I am on a gentle slope, so I should move slowly, I should move fast and I am now on a very fast loop, so I should move slow. So, this having this 1 eta is not working for every point on the error surface right, for everywhere on the error surface, is that clear ok so, ok. So, we will see such algorithms soon where we try to adjust this learning rate ok.

(Refer Slide Time: 02:06)

Tips for initial learning rate ?

- Tune learning rate [Try different values on a log scale: 0.0001, 0.001, 0.01, 0.1, 1.0]
- Run a few epochs with each of these and figure out a learning rate which works best
- Now do a finer search around this value [for example, if the best learning rate was 0.1 then now try some values around it: 0.05, 0.2, 0.3]
- Disclaimer: these are just heuristics ... no clear winner strategy

Mitesh M. Khopra CS7015 (Deep Learning) - Lecture 5

Now, here are some tips for the learning rate. So, how do you, if you are just going to deal with this gradient descent or nag or momentum, how do you adjust these learning rate. So, how do you fix a learning rate? So, a learning rate is typically something known as a hyper parameter. So, why is it called a hyper parameter? So, what are your parameters?

Student: Which I learned.

Which I learned using the objective function; eta is not a part of the objective function, you are not computing radians with the spectra, it is a hyper parameter. So, you will try to tune this hyper parameter. So, what you will do is, in practice you could try these different values on a log scale ok. Next what will you do? Run this all these for a few epochs, note down the dash, just note down the loss function right.

So, done all of these with different learning rates, for say 5 epochs, you will get some loss right. Now which one will you pick? The one which led to the maximum decrease in the loss, I will keep that learning rate and now what you will do? You just stick to that I started off with a dash scale.

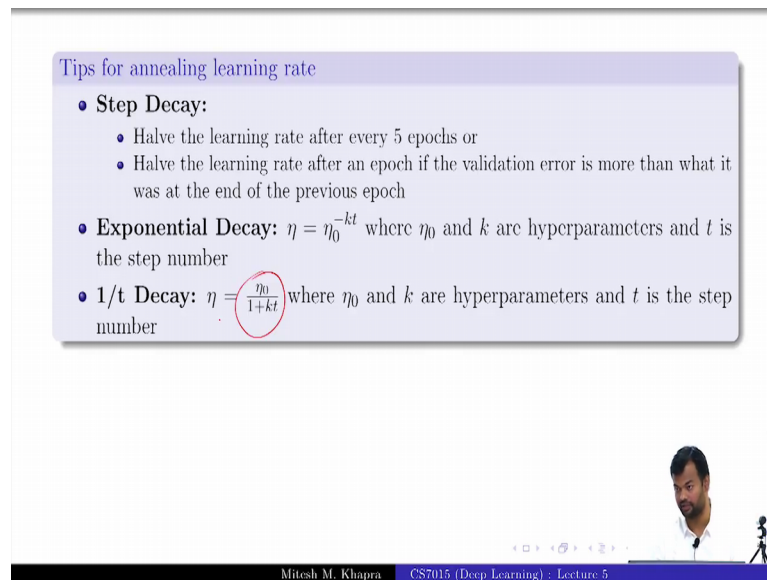
Student: Log scale.

Log scale now what will you do ok. So, now, run it for a few epochs. Figure out which of these learning rates on the log scale works well. Now do a finer search around the best learning rate that you discovered right. So, say 0.1 was the best on the log scale. So, now, look at 0.2, 0.3, 0.4, 0.5, look at values around it and see which one works better right.

So, this is how you will tune the hyper parameters; otherwise there is a very wide range right, if you put tune from 0.0001 to 0.1, there are just too many values to consider right. So, we will have to do this log scale and then a linear scale, will that make sense.

These are just heuristics, there is no guarantee that will always work or which of these are clear winner strategy, but you have to try this. So, tuning a learning rate is an important part, when you are working in deep learning right. So, at least when you are working with gradient descent or nag a momentum based gradient descent.

(Refer Slide Time: 04:02)



Tips for annealing learning rate

- **Step Decay:**
  - Halve the learning rate after every 5 epochs or
  - Halve the learning rate after an epoch if the validation error is more than what it was at the end of the previous epoch
- **Exponential Decay:**  $\eta = \eta_0^{-kt}$  where  $\eta_0$  and  $k$  are hyperparameters and  $t$  is the step number
- **1/t Decay:**  $\eta = \frac{\eta_0}{1+kt}$  where  $\eta_0$  and  $k$  are hyperparameters and  $t$  is the step number

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 5

Now, here some tips for annealing the learning rate ok. So, there is something known as step decay. So, what you can do is, halve the learning rate after every 5 epochs. Can you tell me the intuition for this? What do you expect after 5 epochs? That you have moved enough and now you are closer somewhere to the solution. So, if I closer to the solution, if I closer to phoenix market city you want to move fast or slow?

Student: Slow.

What will you do?

Student: (Refer Time: 04:32).

Decrease the learning rate right. So, after every 5, now this is again what is so sacrosanct about 5, it is just a magic memory, so this is again hyper parameter. So, you could fix some number of epoch and after these I will just have the learning rate ok. Now this second one is what my favourite is and I typically use this, what I do is? I compute the loss after epoch  $t$ , I run epoch  $t$  plus 1, I compute the loss again. If the loss has increased what will I do? I will just throw away all the updates that I have made in this epoch; I will decrease the learning rate and again learn, again start this epoch. What do I mean by throw away all the updates?

Student: (Refer Time: 05:17).

So, after epoch  $t$  I will save my model, I will save all the  $w$  values that I have computed and I will let it run for one more epoch. After this epoch if my loss function actually increases, I reload this model which I had saved, half the learning rate and then run this epoch again. Does that make sense ok. So, I have run till epoch  $t$ , I have some values of  $w$ s and  $b$ s, I will save this values, I will just save it as a numpy array ok

Now, I will with the same learning rate that I have been using so far I will run the epoch  $t$  plus 1 and I get some new values of  $w$  comma  $p$  right. I will plug this into the loss function; I will plug this into the loss function I will get two loss values. If this loss value is greater than what I was at the previous time step; that means, things did not work out well in this particular epoch.

So, I will throw away all these updates. I will just reload the model which I had saved, I will just start from where I was at epoch  $t$ , I will decrease the learning rate I will make it half and run this epoch again right and hopefully now I should do better, because there is something. I am just making a hypothesis that the reason I did not get to a better loss function, was because my learning rate was not adapting to it right.

So, I will just half the learning rate, because this solution was good, this was a low loss function. I just want to be something around it; I do not want to make any drastic steps. So, I will just half the learning rate from there. So, then you not see this drastic change that, your loss function should not improve. So, first of all local minima is known problem in deep neural networks right.

So, what happens is that, in deep neural networks you do not have something which is like a neat convex function as your loss function right, it is a non convex function which means there is no one unique minima, there could be several minima and there are several analysis which show that a lot of these minima are equivalent ok. So, in practice these are the things that you do. Either once you reach a minima you just stay there. The second thing that you could do is, you have trained your algorithm trained your parameters for say 100 epochs and you have stopped now

Now, again go back and start with a different initialization, you started with some  $w$  naught  $b$  naught and you have reached to some solution keep this solution. Now start with a different initialization; that means, if you look at your  $wb$  plane, you have started

from some other point; that means, you started from some other error location right and run this algorithm again, and see if you reach a different minima.

So, the only thing you, the way you counter this is, you just try different stochastic things right, should try to start with 10 different initializations every time reach a minima and then at the end select the lowest possible of these. Did this make sense to most of you, how many of you got this oh cool ok? I thought I was just rambling, but yeah fine. Does that make sense, to you at least ok, does it fine

Yes a local minima is a severe problem in lot of deep learning optimization and typically people get away by that, by just picking up one of these minimum fine. Now the other thing is you could use exponential decay, where with each time step you just keep decreasing your learning rate right. And if this case 2; that means, at every time step you are halving the learning rate, so you just get with something like this ok.

But the reason I do not like this is that you have one hyper parameter which is eta which you are trying to tune and now to tackle that problem you have introduced one more parameter which is k, hyper parameter which is k right. So, it becomes harder to tune that now, and there is a similar thing which is  $1 - \frac{1}{t^k}$ , where you try to use this formula to decay or learning rate ok. So, both of these, I typically do not use in practice I use the second one, I prefer the second one ok.

(Refer Slide Time: 09:11)

The slide is titled "Tips for momentum" and contains the following text:

- The following schedule was suggested by Sutskever *et. al.*, 2013

The formula shown is: 
$$\mu_t = \min(1 - 2^{-1 - \frac{t}{250}}, \mu_{max})$$

where,  $\mu_{max}$  was chosen from  $\{0.999, 0.995, 0.99, 0.9, 0\}$

Handwritten notes on the slide include:

- $\mu_0 = 0.5$
- $\mu_{250} = 0.75$
- $\mu_{750} = 0.875$
- $\delta \propto \mu \cdot \eta$
- $\delta \propto \mu \cdot \eta$
- $\delta \propto \mu \cdot \eta$

The slide also features the NPTEL logo and the text "Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 5".

Now, tips for the momentum can you make sense of this, you just stare at it it looking just come back ok. Let us see what happens at  $t$  equal to 0, this becomes 0.

Student: Log 1.

Log 1 is 0, this is  $2^{-1} - 0$  which is just  $2^{-1}$  which is 0.5. So, what is your  $\mu$  at  $t$  equal to 0.5 ok. Does that make sense? Is it fine with everyone or is it confusing; no  $\mu$  max is typically this, let us assume  $\mu$  max is this fine.

Now, what happens at time step 250? this is  $2^{250}$ , so this becomes 1,  $1 + 1$  is 2, the best thing that you learn in this course log of 2 is 1, so this become  $2^{-1}$

Student:  $2^{-2}$ .

Minus 2 which is 0.25. So, what is this?

Student: 0.75.

0.75 ok, let us do one more I had  $t$  equal to 750  $1 - 1/8$ , so that is what is going to be right ok. So, then what is happening as my time steps are increasing, what is happening to, what is happening? I am having more and more faith in the history or the current gradient. What am I increasing? Actually I have made a mistake, actually this is  $\mu$  is  $\gamma$  there is not we did not use  $\mu$  anyway what you guys just went along. So, this is  $\gamma$  actually right that was a momentum term that we had. So, as a number of time steps is increasing, my  $\gamma$  is increasing; that means, I am having more and more faith in my.

Student: (Refer Time: 11:49).

No history. Learning rate is  $\eta$ , momentum is  $\gamma$ . So, its  $\gamma$  into update  $t - 1$  and  $\eta$  into gradient at the current time step right and here  $\gamma$  is actually equal to  $\mu$ . Is there any more confusion that I can add? ok. So, when I say  $\gamma$  I mean  $\mu$  and so that is how it is. So, as I am increasing the number of time steps I have more and more faith in the history; that means, I do not want to now get distracted by this one update which I am making right, I want to go by the history, and I am not increasing this  $\gamma$  or  $\mu$  indefinitely, I am capping it by a max right. Max I will have this much



faith which is 0.999 in the history. Does that make sense? This is again just a heuristic; do not worry too much about it. So, that is how it is.