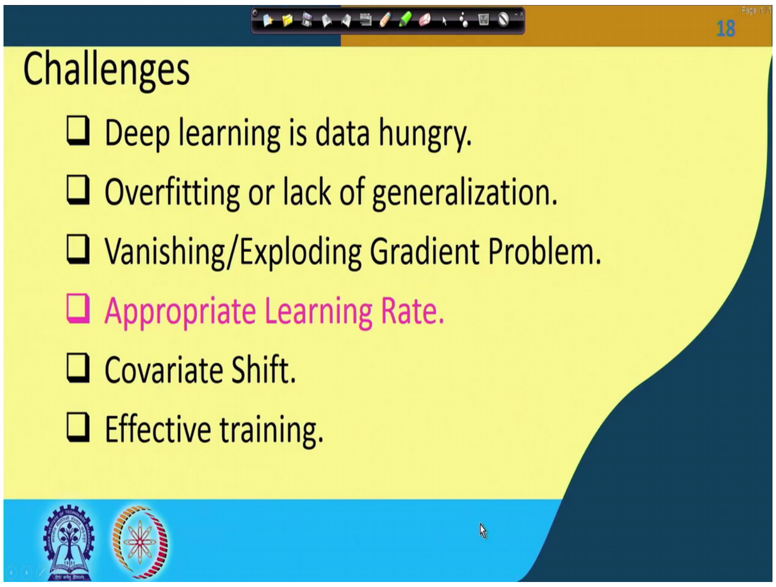
Deep Learning

Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 43
Optimisers: Momentum and Nesterov Accelerated Gradient (NAG) Optimiser

Hello, welcome to the NPTEL online certification course on Deep Learning. Since, our last class, we were discussing about the various challenges which are faced during the training of deep neural network.

(Refer Slide Time: 00:45)



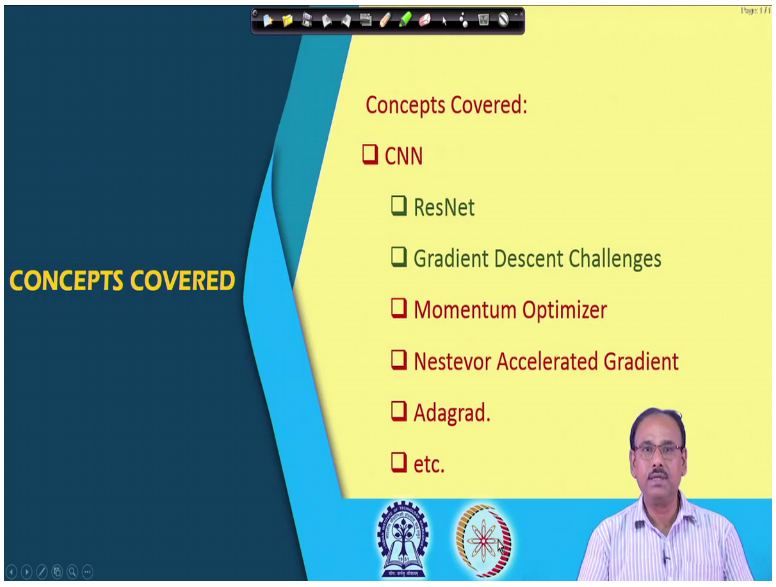
18

Challenges

- ❑ Deep learning is data hungry.
- ❑ Overfitting or lack of generalization.
- ❑ Vanishing/Exploding Gradient Problem.
- ❑ Appropriate Learning Rate.
- ❑ Covariate Shift.
- ❑ Effective training.

So, one of the challenge that we are currently discussing is that how do you choose the appropriate learning rate or what will be the rate of learning or what will be the rate of convergence of the learning algorithm. And, the learning of course what we are discussing about is the gradient descent approach or the stochastic gradient descent approach, and more particularly the gradient descent approach which is considered is what is known as mini batch gradient descent approach. So, we are talking about this deciding about the appropriate learning rate, so that your back propagation learning algorithm becomes more efficient.

(Refer Slide Time: 01:28)

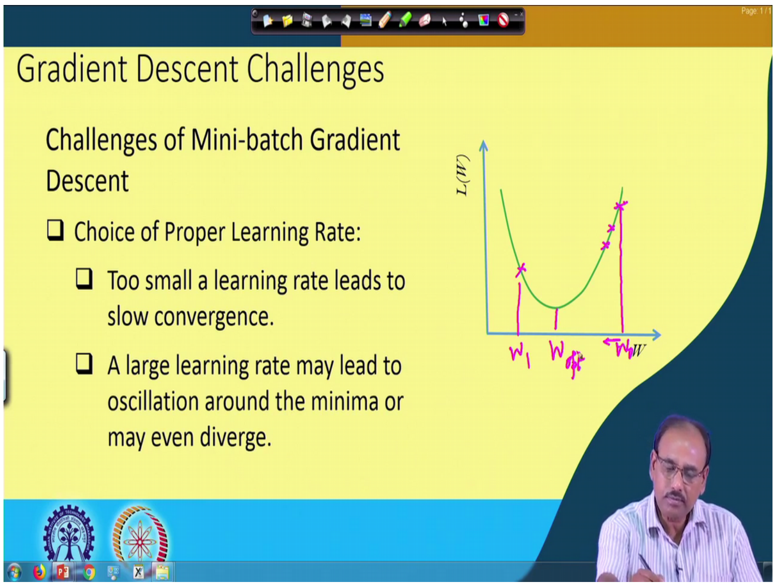

CONCEPTS COVERED

Concepts Covered:

- ❑ CNN
  - ❑ ResNet
  - ❑ Gradient Descent Challenges
  - ❑ Momentum Optimizer
  - ❑ Nestevor Accelerated Gradient
  - ❑ Adagrad.
  - ❑ etc.

So, in the previous class, we discussed about that, what are the different challenges that you face in that gradient descent algorithm itself.

(Refer Slide Time: 01:43)


Gradient Descent Challenges

Challenges of Mini-batch Gradient Descent

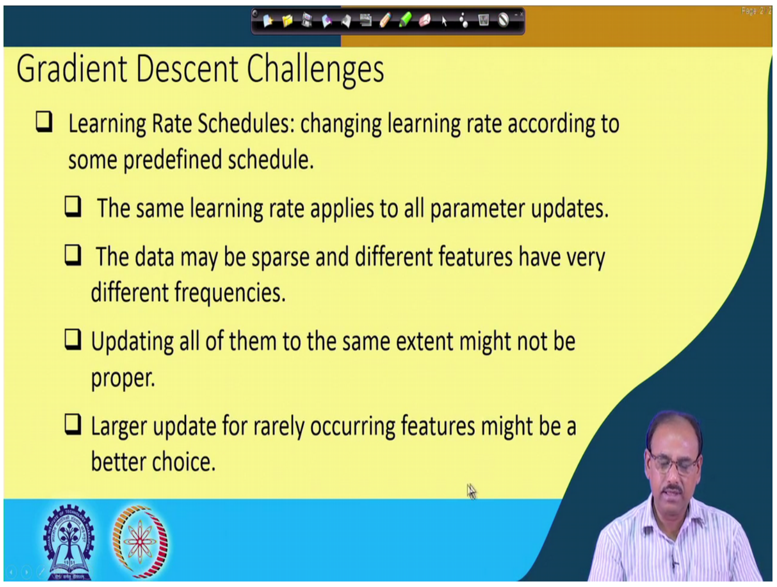
- ❑ Choice of Proper Learning Rate:
  - ❑ Too small a learning rate leads to slow convergence.
  - ❑ A large learning rate may lead to oscillation around the minima or may even diverge.

So, when I talked about this challenges, one of the challenge of course as we said that how do you decide about the learning rate, or when you are updating the weights where vectors of the parameters of your classifier what is the step size that has to be considered for updating the parameters. If the step size is very large, in that case there is a possibility that while updating the parameters, the algorithm will simply jump over the minimum error location; or when the step size is very small it will take a large number of iterations to reach the minimum, so that can be explained with the help of this particular diagram.

Say if I assume that initial weight vector say W 0 has been set somewhere over here then you find that here the gradient, if I follow the gradient descent approach, then this weight vector or weight has to be modified or updated in this direction for reduction of the loss. Now, here if the updation step is very small, then at the next point the weight will be somewhere over here. So, at the next instant, the weight will be somewhere over here; at the next instant it will come over here and so on. So, as a result it takes large number of iterations for convergence.

Whereas, if the step size is very large, then it is possible that at the next moment your W 1 will come somewhere over here, so this becomes the position of the W 1. So, as a result you are jumping over this minimum location this might be your optimum weight vector which will minimize the error. So, choice of proper learning rate or the proper step size for updation of the weight is very, very important.

(Refer Slide Time: 03:52)



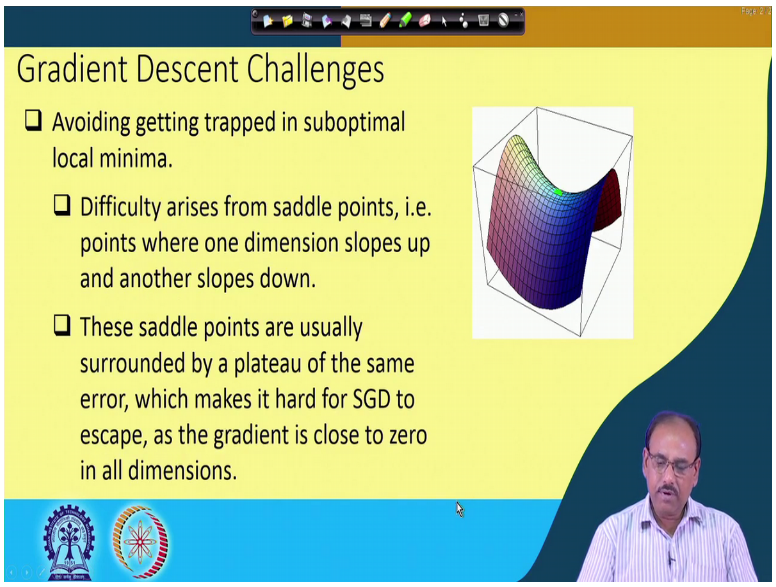
Gradient Descent Challenges

- Learning Rate Schedules: changing learning rate according to some predefined schedule.
  - The same learning rate applies to all parameter updates.
  - The data may be sparse and different features have very different frequencies.
  - Updating all of them to the same extent might not be proper.
  - Larger update for rarely occurring features might be a better choice.

The second challenge in this gradient descent approach that we have said is like this. It might be possible that I can have a predefined schedule that how the learning rate or the step size will change over the number of iterations or over the epochs. But if you do that, it is the same step size which will be applied to all the parameters or all the components of your weight vector which may not be very appropriate, because the gradient may be very large with respect to certain parameters or it may be very small with respect to some other parameters.

So, the parameters with for which the gradient is very large for those parameters, I may like to have smaller step size and the parameters for which the gradient is quite small, I may like to have a larger step size. So, that is very difficult to define beforehand, so that means, predefined schedule of learning rate is extremely difficult. And it may be possible that you may have to update or you may have to tune the learning rate on the fly as you go on learning over different iterations. So, that is not possible if I go for predefined schedule of the learning rate.

(Refer Slide Time: 05:12)



Gradient Descent Challenges

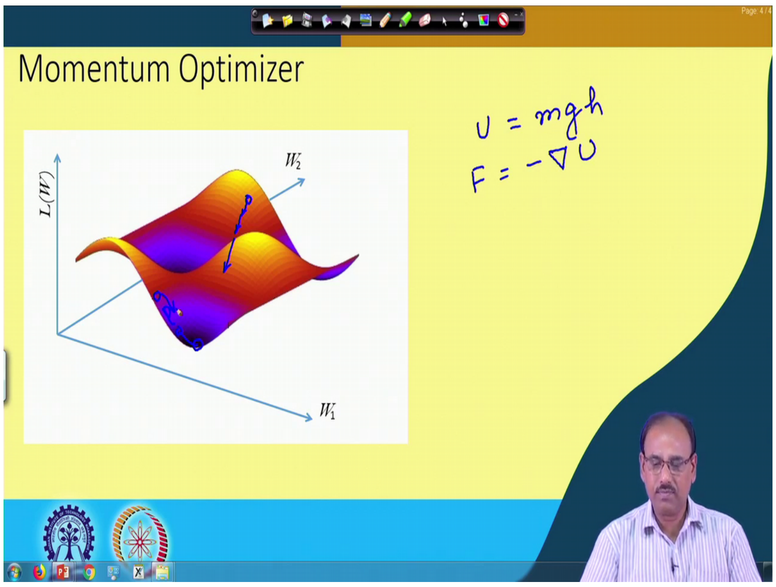
- Avoiding getting trapped in suboptimal local minima.
  - Difficulty arises from saddle points, i.e. points where one dimension slopes up and another slopes down.
  - These saddle points are usually surrounded by a plateau of the same error, which makes it hard for SGD to escape, as the gradient is close to zero in all dimensions.

The other kind of problem that you face is that while learning when the algorithm comes across the saddle points. Saddle points are nothing but the points where in one dimension the slope is in the positive direction, whereas in the other dimension the slope might be in the negative direction. So, an example of saddle point is over here. So, this is what is in the saddle point, here I have a saddle point. So, here you find that in this direction the slope is positive it is sloping up; whereas, in this direction, it is sloping down.

So, in such cases the gradient descent approach finds it very difficult to navigate that reason being such that saddle points are surrounded by plateau where on all the points on the plateau the error is almost same that means, the gradient vanishes and as the gradient is almost zero, the algorithm does not learn anything. So, these are the different challenges that you face when you use the gradient descent approach. So for that what to overcome this problem, what we have to think of the different approaches by which the gradient descent algorithm can be optimized further.

Or in other words, what we would like to have is an approach or the algorithms by which the gradient descent or the back propagation learning algorithm can be more efficient. So, one of the approach that can be used for making the gradient descent learning algorithm very efficient is momentum optimizer. So, today we have we will try to see the different types of optimizers one of them is momentum optimizers, the other one is Adagrad and we will talk about other different optimizing techniques in our subsequent lectures.

(Refer Slide Time: 07:21)


Momentum Optimizer

$U = mgh$

$F = -\nabla U$

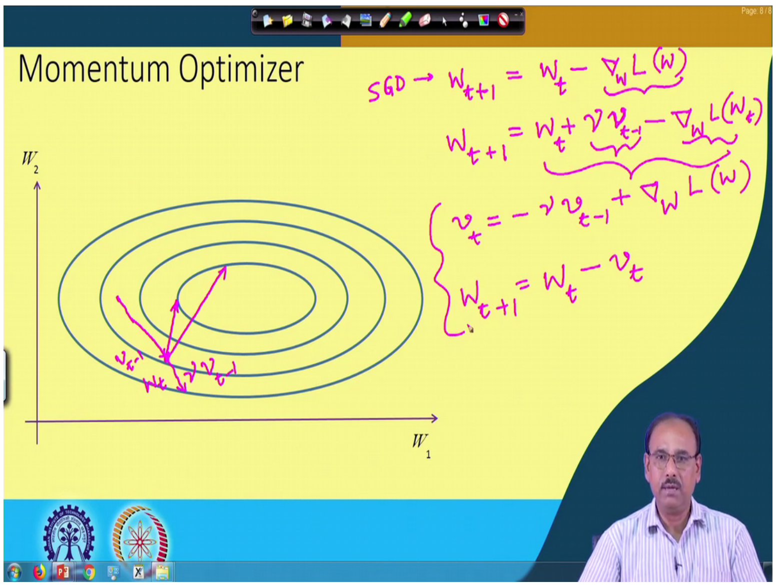
So, first let us see that what is this momentum optimizer. You find that this gradient descent algorithm I can have analogy of this with an example that I put a ball on a hilly terrain, initially with an initial velocity of 0. So, it is something like this that suppose I put a ball over here, let me change the color ok. So, suppose I put a ball somewhere over here with an initial velocity which is 0, and this ball has got certain height and because of this it has got an initial potential energy.

And the potential energy is given by U is equal to m g h, where obviously, m is the mass of the ball, g is the acceleration due to gravity, and h is the height of the ball. And there will be a force which will be acting on the ball which is negative of the gradient of the potential energy that means, the force which acts on the ball is equal to minus gradient of the potential energy which is U. And under influence of this force the ball starts sliding along this hilly terrain or along the surface.

So, it moves like this. And, as it moves down this hilly terrain, it gains momentum. And subsequently the ball reaches this minimum point which is the plateau and because it has a momentum, it will overshoot the plateau and will start moving in the other direction. And how much it will move in the other direction that depends upon how much is the momentum it has gained when it has reached the minimum point, and also what is the opposing friction or what is the damping force acting on it.

So, while going on the other side, it will come to rest at certain point of time. And from there again it will start coming back to the minimum positions, so it will perform a number of oscillations around the minimum position before it settles at the bottom of the surface. So, our gradient descent approach that, we are discussing that can be compared with this particular analogy ok.

(Refer Slide Time: 10:01)



Momentum Optimizer

$$SGD \rightarrow W_{t+1} = W_t - \underbrace{\nabla_W L(W)}$$

$$W_{t+1} = W_t + \underbrace{\underbrace{\nu v_{t-1}} - \underbrace{\nabla_W} L(W_t)}$$

$$\begin{cases} v_t = -\nu v_{t-1} + \nabla_W L(W) \\ W_{t+1} = W_t - v_t \end{cases}$$

So, this algorithm works fine as long as my surface is and well behaved surface that is along every parameter in every direction, the curvature or the slope of the surface is almost same. But you think of a situation where I have a surface which is an error surface, where the curvature is very small in certain direction and it is very high in some other direction. So, if it is so, then the gradient or the direction of the gradient in the direction where the curvature is high will be very high, and in the direction of the curvature is low the component of the gradient in that direction will be very low.

So, in this figure, what I have shown is it is the planar projection of an hyper ellipsoid or the surface I can consider to be an hyper ellipsoid, and in three dimension it is an ellipsoid. So, if I take a projection on the plane, I assume that my vectors are two dimensional vectors having components W 1 and W 2. And every closed contour on this diagram, this, they represent loose eye points of equal energy ok.

So, given this kind of situation, now you find that if I my initially the weight vector is somewhere over here or if I put a ball somewhere over here as we said before that under the influence of the gradient of the potential energy which is nothing but the force on it acting on it. It will start sliding down the surface. And the minima of the surface is somewhere over here somewhere over here I have the minimum.

So, this gradient force which is acting on this will have two different components, one component is in the vertical direction and other component is in the horizontal direction. And as we have said that, the curvature of the surface is very high in the vertical direction compared to the curvature of the surface in the horizontal direction. So, as a result the component of the gradient which acts on this particle in the vertical direction is very high compared to the component of the gradient in the horizontal direction. As a result the force which is acting on this ball will be in this particular direction.

Now, assuming that it comes to rest over here, from here again you compute the gradient it moves in this direction as again the component in the vertical direction is higher than the component in the horizontal direction. From here again, it will move downward direction like this it will move in the upward direction like this and it will try on oscillating and you find that because of this to and fro oscillation in the vertical direction which is very high in the vertical direction, the number of such iterations the algorithm will take before it converges at the minima will be very high.

So, this is what you have in case of a typical gradient descent algorithm. And you find that you need large number of iterations because the gradient in one direction or gradient in that direction of W 2 is much larger than the gradient in the direction of W 1. So, I can avoid this problem if I bring in a concept of momentum. So, the concept is something like this that again I assume the ball is over here, its gradient force acting on this, so the ball comes somewhere over here.

Now, at this location the gradient working on this ball may be in this direction, whereas if I also consider the component of the momentum that is a force due to momentum of the ball which is in this direction. So, if I consider both this gradient force as well as this momentum force to find out what will be the net force which is acting in this ball. So, the net force if I take it which is sum of these two forces, the net force acting on the ball will be in this direction.

And I assume that the ball will move in that direction of this resultant force. So, here you find that instead of you are updating the weight vector in this direction, you are updating the weight vector in this direction. So, as a result you are moving faster towards the minimum location, and that is what is the effect of considering the momentum along with a gradient descent.

So, what is the impact of this in our algorithm? So, if you remember the gradient descent algorithm works like this, that I want to find out the weight the parameter at time t plus 1, and this parameter at time t plus 1 is obtained from the parameter of the weight at time t minus gradient of the loss function, where loss is a function of the parameter or the function of the weight vector W. And this gradient has to be taken with respect to my parameter vector which is W. This is what is my normal stochastic gradient descent algorithm.

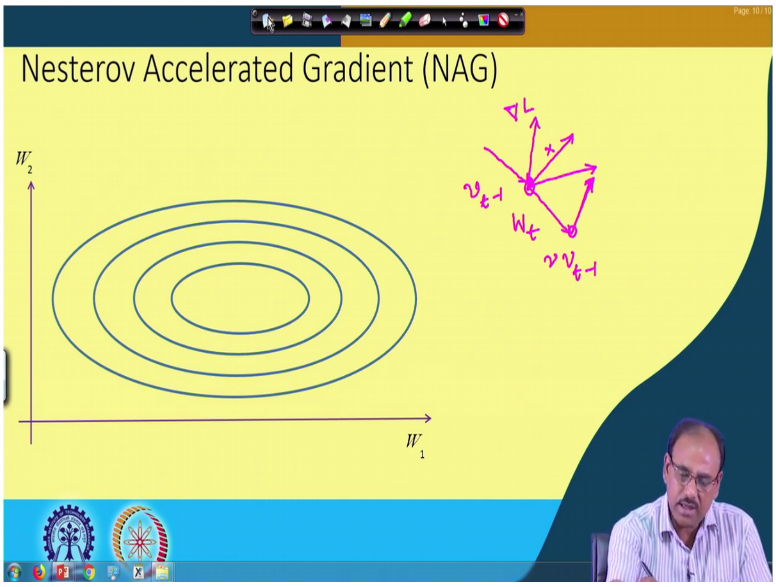
And now what I do is in addition to this I want to add a momentum term right. So, I assume that at time instant t that is W t is somewhere over here this is the location of W t. And it comes to W t from location W t minus 1 and with a gradient vector which is v t minus 1. So, this was somehow W t minus 1 from W t minus 1 it comes to location W t with a resultant force under the influence of a resultant force v t minus 1 resulting on this. And at this location W t, I have two forces acting on it once is the gradient force which is this gradient of L W with respect to W, and other one I consider is the momentum force which is some gamma times v t minus 1.

So, what I am adding is I am adding this momentum term to this gradient force. So, as a result, W t minus 1 the weight updation equation will now be W t plus 1 is equal to W t minus this gradient term which I had before it remains as it is L W t. In addition to this, what I am adding over here is the momentum term which is nu v t minus 1. So, this is the momentum term, and this is the gradient term. So, under the influence of these two now the net update direction of the weight vector will be in this direction instead of in this direction, and that is how your gradient descent approach with momentum improves the rate of convergence or makes this back propagation learning more efficient.

So, this particular equation I can put in another form, I can write a gradient is v t in terms of v t minus 1 which is minus nu times v t minus 1 plus gradient of L W with respect to W. And after I write this my weight updation equation can be W t plus one becomes W t minus v t. So, these two taken together becomes the gradient descent or the vector equation learning algorithm considering the momentum effect. And as a result of this, you find that if I put to the two figures side by side, on the left hand side, the weight addition sequence without s d without the momentum term is shown over here.

So, these are the weight updation sequences with momentum term you find that the weight updation sequence will be something like this. So, when I consider this momentum term the gradient descent algorithm or a stochastic gradient descent algorithm becomes much more efficient. Now, there has been another modification on this momentum approach or the stochastic gradient descent or with momentum that has been suggested which improves this momentum optimizer to a certain extent and that is what is Nesterov accelerated gradient approach or known as NAG.

(Refer Slide Time: 20:09)



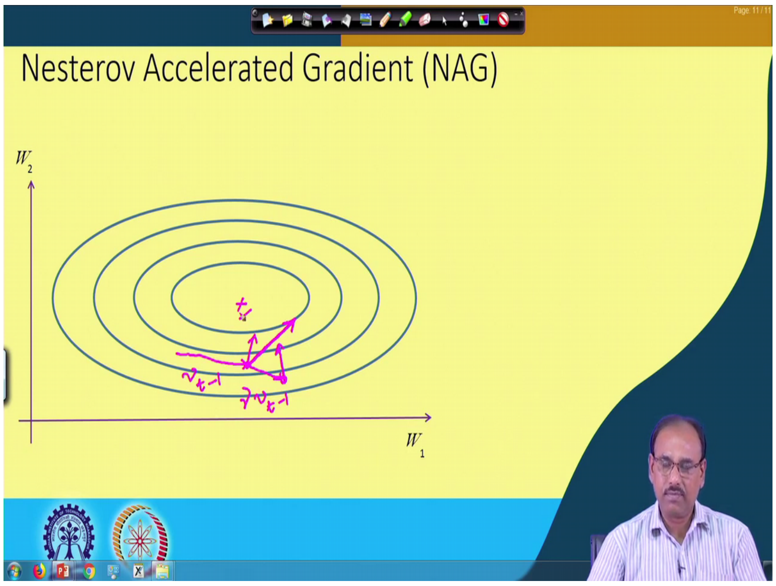
Nesterov Accelerated Gradient (NAG)

So, what is the NAG? You find that in every case with momentum, what I am assuming is same i at time t, so I have W t somewhere over here, I come to W t with a momentum with a previous force which is equal to v t minus 1. And due to this I have a momentum term which is nu times v t minus 1, I have the gradient at this location which is delta L. And under influence of these two, the net displacement or the weight updation which will be done parameter updation will be given by the sum of these two vectors which is in this direction.

Now, how this acceleration can be done or the gradient can be accelerated is this, if I know beforehand because I know what is my momentum. And I assume that due to a effect of this momentum what will be my position after effect of this momentum that is where I am going to lead over here. And if I know that what is going to be my position because of the momentum effect in future, then instead of considering the gradient at this location I can find out what will be the gradient at this particular location.

So, if I can do that, then suppose I will have the gradient vector which is falling in this direction, then I can have an update instead of this I can have update location update direction, which is sum of this momentum term and the gradient computed at the future location. So, this is what is my look ahead gradient, I am computing the gradient beforehand. So, I call it and look ahead gradient. So, you find that if I modify or if I update the parameters using this look ahead gradient, in some cases this look ahead gradient approach that may even improve the gradient descent algorithm further.

(Refer Slide Time: 22:31)



Nesterov Accelerated Gradient (NAG)

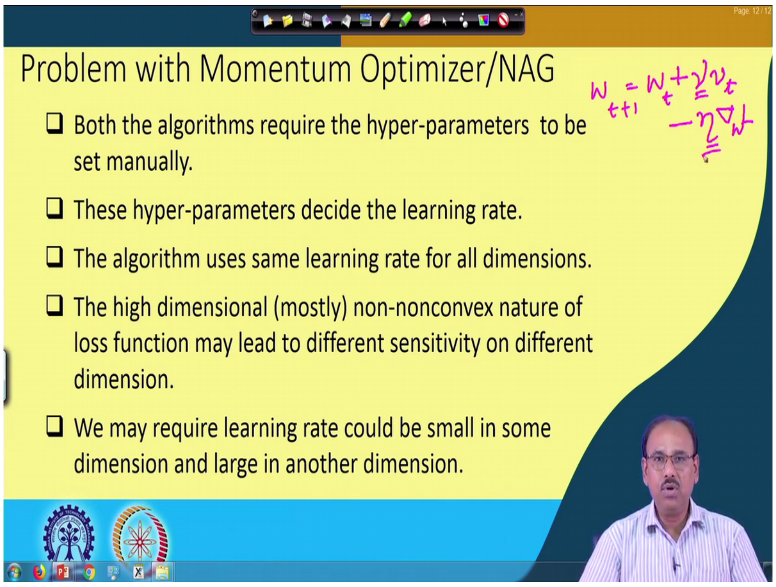
So, just to illustrate this let us assume that we have say at time W t that W is over here or my parameter is over here. And I have come to this parameter following of v t which is given this. So, this was my or v t minus 1 which was in this direction. So, at this location, I have nu times v t, v t minus 1, I have gradient over here which say acts in this direction and the under the influence of these two, I will have a net update which moves over here.

Now, with this accelerated gradient, what I can do is I can assume that the position will be somewhere over here in future, and I now compute the gradient vector at this location. So, if I compute the gradient vector at this location I will my a great updation will be over here. So, you find that this is the location of the minimum loss. So, using this in Nesterov of accelerated gradient approach, you are moving faster to the minimum. And if you do not this accelerated gradient approach, you are moving in this direction.

So, now, you will need more number of iterations or more number of epochs before you come to the minimum. So, this NAG or Nesterov of accelerated gradient helps in improving the gradient descent algorithm further. So, till now we have discussed about two approaches, one is the gradient descent with momentum and next the gradient descent with a modify the momentum approach where we are computing the gradient at a future location which is known as which we are calling as look ahead gradient operation.

I am computing the gradient at a location, where I can guess that that will be my location in future and this approach that is NAG or Nesterov accelerated gradient approach improves the performance of the gradient descent algorithm even further.

(Refer Slide Time: 24:54)



Problem with Momentum Optimizer/NAG

- Both the algorithms require the hyper-parameters to be set manually.
- These hyper-parameters decide the learning rate.
- The algorithm uses same learning rate for all dimensions.
- The high dimensional (mostly) non-nonconvex nature of loss function may lead to different sensitivity on different dimension.
- We may require learning rate could be small in some dimension and large in another dimension.

$$W_{t+1} = W_t + \underline{\underline{\nu}} v_t - \underline{\underline{\eta}} \nabla_W L$$

So, these are the two approaches that can be done, one is the momentum based gradient descent, other one is accelerated gradient descent. However, you find that in both of these cases I need a hyper parameter. So, the hyper ammeter that we have said is in case of this gradient descent what I had is W t plus 1 is equal to W t plus some nu times v t minus some eta times gradient of L with respect to W. And this nu and eta these are the hyper parameters which determines how fast I am going to move towards the minimum loss location, and these are the hyper parameters.

And in case of both this momentum optimizer as well as NAG or Nesterov of accelerated gradient, I require this hyper parameters to be set manually which is difficult thing. And as this hyper parameters decent the learning rate, values of this hyper meta parameters are very very important. Not only that this algorithm uses the same learning rate or the same step size for all the parameters for W 1, W 2, I have the same value of eta, I have the same value of nu.

And as we have discussed before that may be I mean and that is what we have seen also that in the vertical direction, your gradient is much more than the gradient in the horizontal direction. So, if I can have my step size in the vertical direction lower than the step size in the horizontal direction, the learning will be much more efficient, but which is not done with this momentum based optimizer or even this Nesterov of accelerated gradient approach. So, that is another problem which is faced in momentum optimizer as well as in NAG.

You also find that the high dimensional and mostly non convex nature of the loss function that may look to different sensitivity or different dimensions. So, as a result, I may like to have different learning rate for different parameters which is also not possible using this momentum based optimizer or NAG. So, I will stop here today. In our next lecture, we will try to see some other algorithms where these concerns that having the same learning rate or avoiding having the same learning rate for all different parameters that can be improved further.

Thank you.