Modern Computer Vision

Prof. A.N. Rajagopalan

Department of Electrical Engineering

IIT Madras

Lecture-16

So, last class right we saw back propagation right and we saw that you know you could have deep you could have a deep network and still be able to propagate right do propagate the error backwards and then update the weights update the bias and all that we saw the matrix vector form for doing that. Now, there is just one thing that that I that I think I would want to tell which is about about you know see we have not yet said you know said anything about how we how we take these samples. So, for example, you know. So, for example, in the in the last one that we had I mean it was I think what was it it was just a mean square error kind of thing right. So, we had like summation y i hat minus y i square and then we said i going from 1 to 2 and then we scaled it by half right. Now, ideally right you would have several samples right think of a case where you have images and you want to denoise them.

So, y i will be like one image right and then y i hat will be your estimated image and then you want to find out the mean find out you know sort of the even square error right between the two and, but then you would have to do it over several examples right you cannot learn with just one example or something. So, normally what happens is you will have something like y i hat minus y i square and then let us say right i going from 1 to m and then you can have another summation where let us say where let us say right I mean you know. So, we can say maybe right y i j and then y i j where let us say j is the j is the j is the right you know sort of example number. So, you can have like j is equal to 1 to some some right l number of examples and right I mean basically what you might have to do is do is for every example right you might want to and then you might want to scale this by you know 1 by l m or something right and you would want to want to make sure that over all the examples you get eventually an error which is which is small enough right.

Now, it could be may it could be whatever it could be one dimensional it could be even speech data whatever it is right. So, what you do is you take. So, in a sense right you might think that what probably we will do is you know we will push all the examples and then right before we even do a gradient update we would want to compute this cost over all the examples and then one then once you have once you have once you have accumulated right that kind of an error then we will then we will update. But then what happens is that becomes you know too slow a process because of the fact that you know you may have a

million examples for example right and you know you would have to first of all you know push all of them through and then because of the fact that right you have got so many examples out there even the you see is right movement is not no no will not be fast enough. So, the so there are kind of right 3 ways of doing gradient descent only one is typically followed, but then I just thought right I will just talk about talk about it all 3 of them right what they are.

This we could have done before, but right at that time I left it, but I think right now we will just take one you know quick look at that. So, let us look at the first one right which we call batch GD right. So, this batch gradient descent is like batch GD is like no. So, the entire training set the entire training set it is used in each you know iteration and that means before you even update. So, always right let us kind of go back to that you know to that update equation which is like whatever you know your theta t plus 1 or n plus 1 whatever you had is equal to theta t minus alpha let us say gradient of gradient of L dou L theta with respect to theta evaluated theta equal to theta 2 right.

This is this is always the case, but now, but now it looks like right this this update which we are going to do theta could have both your weights and the bias. So, the update right in batch GD what will happen is you will have to you will actually compute this L theta right here is your L theta for example right this is your L theta. So, you compute your L theta over all the examples okay, but then right this is not this is not you see right normally followed because it is just you see right I mean even kind of what is a computationally you have to do it over a million examples before you even you know right make the first update. And the you know second thing is storage, memory and also the fact that you do not you do not want the movement to be so slow. The other extreme right that you can do is what is called a stochastic gradient descent or SGD what is called a stochastic gradient descent.

Of course, you know this is also come to be I will tell you that I mean so right now just kind of take it for granted that we call the stochastic gradient descent or SGD and you know so right here it is the other extreme. So here the weights are updated, the weights are updated, updated means every right iteration whenever we say update we mean after the iteration, after computing gradients for every training sample, after computing gradients for every training sample. That means you know so in this equation you will probably take this one example right compute the error and then update the weights right that will be like you have to say right iteration. But then this is this is too oscillatory because every example right will try to will try to know take you from here to there and then right you would not have sort of you know a direction because one guy will swing it one way another example will swing it the other way. So really this extreme is not something right that you would want to use but yeah I mean right nothing stops you from doing it.

The weights are updated after computing gradients for each training example, for each training example. So the drawback of the first one is that is that it is too heavy right in the sense that the right inertia is too high. So this is kind of slow okay. This can give so no kind of rate I mean what do you say random courtesy directions right because the fact that every example will try to will try to swing it in a sort of a different direction. So again right not the one that is typically used.

So you would not use this typically right in any of your implementations not use this typically what you use is what is called you know mini batch, mini batch GD gradient descent okay and this is what we normally use and in mini batch right what we do is so for example if you had a basket of examples they would then you would actually right break them down into these batches what is called a mini batch and then you would do this loss computation over a batch. So for example right suppose let us say suppose you had not that L right suppose you had done L examples and you take M examples in a batch right then the mini batches will be like you say L by M and you traverse of course the whole training set right you do not leave any example out. You traverse through the whole training set but at a time you take a mini batch of example you can pick the mini batch randomly okay within I mean it is not like you know you have to go in a particular order you can but you make sure that you traverse all the all the examples right and within so you take a mini batch and compute this L theta for that batch. So this L right that I wrote here so that can be that is not that L okay this L is well maybe you can use some other some other L I mean if you wish so this L and that L are not the same okay or if you want to treat L as the L as the number of training samples that you are using okay then yeah that is what I have used there for so this L I will just make this as N or something right where where N is the total number of training samples and N by M will be the will be the number of mini batches right. So this will be the number of mini batches and M is M is the M is the number of examples in one batch in one sort of you know mini batch right.

So therefore M mini so over M examples you will compute the error then you will update the gradient then you will again write I mean you know then you sort of write you know do this in an iterative manner. So of course you know many people refer to this mini batch GD also says GD okay but strictly speaking one should be using it as mini batch mini batch gradient descent okay yeah. So right this is just an aside but just to know that you know when because from now on that iterations we will we will keep kind of see talking about iteration iteration and so on and therefore when you look at this L theta right so you should know that when we talk about an iteration we are we are really really referring to this mini batch sort of a gradient descent which means that not all the examples are involved okay. In an iteration there is only a mini batch of examples involved and when you actually when you actually cover cover all the examples right so that is called an epoch right. One one epoch is like when you have all the training samples covered so that is so

right so now that is like one epoch and normally these networks take several such such as epochs to to to actually train in the sense that for you know convergence to occur it is not like with one epoch right you will be able to attain convergence you will typically require several epochs and that is where all your computations and all come in right whether you have a good processor and so on how much time you have to wait right till you start observing things and so on.

But there are whether certain you see tricks right which you have to do in order to in order to make things move faster okay that is that is the part on this optimization right so what basically people do is they do not simply use gradient descent in this form okay this is the most kind of you know you know sort of simplest this is the simplest form for a gradient descent but what is typically used is not is not is not this form. So what you do is you know you actually introduce certain things right what are all the things okay which you would want to do one is for example can you sort of write accelerate in the sense that you know if you have if you know that you know your slope is rather let us say what do you say you know very shallow right if you have a very very very or it is almost like a flat region right. Now why would you want to write inch at the same rate right I mean when you when you can when you anticipate right that there is that the slope is really less then you could actually accelerate the way you the way you can move forward right I mean it is like saying that I know that that this is a flat terrain so I need not take one small small step to get there right I might just want to accelerate. But then it should be such that it is not just applicable for a flat terrain it should be applicable even in general so that if I can use a past history in order to move forward faster I should be able to use that right. So so this so that is called a momentum based sort of a GD where a momentum is applied so when you when you take the step size right there is accompanying with that there is also over know a momentum which you give so that if you are if you are if you are is the previous sort of a direction and the correct direction are actually aligned right then then you would take a step you know where the previous guy will help you move forward faster right I mean it is like saying that if I if it is like saying that if I had if I had a vector in the original direction which was like this and if what I am you see computing now if it is also right in the same direction then if I add the two right I will go faster forward and and it makes sense right.

But only thing is when you come near the basin you could have problems okay that we will talk about but but but generally right it is a smart idea to actually use this is a momentum to actually to actually right push yourself forward. The other thing is this alpha itself right which is sitting there I mean so we said that is a step size right. Now this alpha is also something which you can optimize and so when we say optimization we really mean tricks that you can employ right in order to have a faster way to kind of get there because otherwise all these things right can be can really slow you down okay so so the and these are not that old by the way okay in fact some of these things are as recent as

2012 and 14 and so on so that way in terms of history right it is not like you know like 20 years old or something and most of them are actually used okay. So the other thing is about the learning rate so this alpha is also called the called the you know learning rate right. So so the idea is that should you use the same learning rate you know or for or you know or for example you know should we have so for example right I mean if you have because in this case theta is actually a vector right.

So so the so the idea is that right can I kind of can I get a fine tune my alpha such that right but you know for for let us say right certain certain elements right I mean I can move faster and for certain right I should not and so on. So this alpha also right does not have to be fixed okay and alpha is alpha as a learning factor right you can also you know tune your alpha okay. So so the idea is that right these are the two things that are mainly done with respect to optimization right and you know which we will walk through now and then and then there are say other aspects which is like you know regularization that we will talk about later. Regularization is all about you know do being able to being able to you know make a network you learn beyond the training examples that you have learned that means avoid sort of overfitting and the other idea behind regularization is that is like you know having a less complex network like I totally read what was what was that to say razor Occam's razor right. So that said that you know among multiple competing hypothesis pick the one that that is the most simplest rate and actually explains the this one situation.

So in the same way right if you just allow your weights to be whatever they can be right then what can happen is all your weights can come on and then you will have a bloated network right where all weights seem to be active and then you would actually be right overfitting and I mean it is like saying that right I mean you know everyone you want them to work instead of that you might say that you might want to have a lean network where you say that I mean you know let me not have all weights act I mean you should have a constraint of the weight that you should not just right let it be unconstrained. Then then actually what you can do is you know then you can actually regularize in the sense that you can have a you can have a network wherein wherein you know the weights come up when they have to right otherwise otherwise that they should not okay. So so optimization and regularization these are these are two two things right which which we should be aware of okay and I am I hope that we will be able to complete both of them today okay. So the first one let me talk about momentum based momentum based GD gradient descent okay. So this momentum based GD looks like this so the weight update equation right becomes something like this.

So the weight update becomes so I will first write this right then I will explain what it means so V T is equal to let us say some gamma V T minus 1 plus alpha this alpha is the same alpha and then gradient L theta theta equal to C theta T and the actual update is like

theta T plus 1 is equal to theta T minus V T. So if you see right I mean earlier earlier all that you had had was this term if you just replace V T right earlier what you had was this alpha this one gradient of L theta but now you have this additional extra term right which is which is this gamma V T minus 1 and V T minus 1 in turn would have come from V T minus 2 and then you know a gradient of L theta at that at that at that particular iteration right which will be like theta equal to theta T minus 1 and so on. So the hope is that hope is that right when you are actually traversing okay right down the curve instead of instead of taking small small steps and trying to trying to get down to this base in so the idea is that if you were to add a momentum right this factor is actually a momentum momentum. So the idea is that if let us say V T minus 1 and this this gradient at the this one the current sort of a time step right if they are actually both you know mutually aligned right then then you would want to sort of add that factor in right in order to be able to able to move forward. Whereas gamma is a number that is typically between 0 0 to 1 okay typically you do not use 1.

So the idea is that you know it is like so it is like right giving an exponentially decaying weightage to the past V T's and as you can see right if you try to write you know if you start with V 0 and let us say you write V 0 is 0 and then we have like V 1 as whatever right. So V 1 will be a gamma V 0 plus something but that term drops off you just have the right hand term then when you go to V 2 right you will get you will get like right gamma V 1 and then and then like gamma V 1 will then actually multiply. So you will see that you will have like gamma square gamma cube and so on it for the past guys. So it is like you know exponentially because right when you are here right you would not want to want to give too much importance to what was what was what is this direction right I mean you know several steps before right. So it is like so this is actually an exponentially decaying exponentially decaying average right exponentially decaying average which actually helps you decaying average averaging right is what you are doing.

What you are doing is an exponentially weighted you know exponentially decaying average. So that so that the right immediate ones take you know take more you know are more get us irrelevant in order to move forward and then the past ones right have you know relatively less role to play and the and the idea is that see the only problem right with this that I mean this looks like actually meaningful thing to do right because it is like saying like I said right I mean even if you even if they let us say right two are not exactly aligned still still it is ok I mean you know, but there is a two are exactly aligned like in a flat surface or something right then you could really move you know much faster. In fact I mean you could add up all the all the right earlier steps and really move forward. But the thing is right the only problem is somewhere here right when you actually when you actually come near the which you do not know right a priori. So if you are if you are coming closer to the kind of minima then then you see what could happen is you could you

could actually you could actually write overshoot I mean because of the fact that you know you have you have a previous gamma right Vd minus 1 which are going to add to this and therefore right what is typically for again that these are all observations right.

So so when this momentum based Gd was implemented people found that there are oscillations when you come when you come right near the minimum. But otherwise you are very fast I mean that means you are able to do your job in much fewer iterations. It is also about how fast you get there right. So it is also about involving fewer number of iterations at the end of the day right. This momentum what will it do it will right you know iterations will become fewer because you are moving faster right.

I mean you are taking larger steps right in a sense instead of taking a baby steps you are taking larger steps to get there and therefore right you expect that in fewer iterations you will get to where you want to be. But the only thing is right when you are kind of when you are kind of hitting when you are when you are closer to the minimum. So there is a right it can have it can you know so so it is what do you say it is known to exhibit. So this is known to exhibit oscillatory behavior known to exhibit oscillations or oscillatory behavior near the minima near minima or minimum. And but but then the then the point is right it is actually used.

There is a variation of this that actually people use which is called a Nesterov momentum. I will just I will just hint as to what it is because like I said it I mean is there anything else that I have to write here. So I think let me just write that when v t minus 1 and delta l theta at theta equal to theta t are well aligned well aligned you get an you get you get added momentum momentum that is why it is called momentum based to well to move faster to move faster. And this is mainly useful in useful in regions with with shallow slopes because right that is when that is when you need to go faster because people write flat regions and so on it is not just to say right it is not that only for flat regions it is useful whenever you have whenever right I mean otherwise otherwise you will have a gradient which is very small and therefore your alpha if it is fixed for some reason actually it is not fixed then if you fix it then your step size is become really small right. So instead of that you could just use a previous history right it is like using history to history to accelerate because you know that that is how you came and therefore it makes sense to use your past history.

Nesterov momentum this goes after a person's name by the way and do I have the year I do not I do not know whether I have the year for this one Nesterov 1983 guy his name is some Urey Nesterov. So this is 1983 work yeah this is old, but there is another thing, but as there are several other things which are relatively new. So here right what what all that he suggests is because of the fact that right you you anyway you know plan to actually

make that step forward which is gamma b t minus 1. So his idea is that instead of actually evaluating the gradient at theta t evaluated evaluated at theta t minus gamma b t minus 1 because you are you are anyway you are anyway right making that step forward right this is already embedded in your distant momentum. Therefore his idea so to sort of what to say so to explain this right in a sense right what it actually means is that especially right if you are okay let us let us kind of go back to this figure right.

So what this means is that if you are actually let me choose a different color. So what this means is that if you are here right and then let us say right this is your a point and from there right you came down to b okay and then and then let us say right you came to c. Now now c is where you are you are sort of I mean a and b were okay you are still far away from from the from the you know minimum, but at c right you are you are sort of c close. Now this is like you know look ahead right you know look before you leap kind of thing okay that is what that is what that is what is effectively means look before you leap or look ahead prior to leaping look before you leap right. So what this means is that so for example if I if I had gone the usual way right what basically might have happened is you know I would have actually incurred a incurred a large step and sort of right ended up here.

Let us say right here is where I end up if I if I do not use an Esterov kind of a kind of a thing right because all that I do is I have a gamma v t minus 1, but then but then my step size becomes so large right that I actually you know end up on the other side. So by saying looking ahead right what they what this let me write down that equation so there is changes in the following way. So instead of so there is only one change right so v t is equal to theta so this all remains the same gamma v t minus 1 plus alpha this is like gradient l theta except that theta becomes equal to theta t minus what is that gamma v t minus 1 right. So so then what this is saying is right so you can actually go ahead and sense as to right what is what is happening at theta t minus gamma gamma v t minus 1 and if there if the slope is already changed right then what will happen is when you do when you when you when you kind of estimate your new v t right that will actually be a reduced step size right because because the sign has changed right otherwise you would have left forward to actually d right. See it does not mean that you do not cross over the minima ok that need not I mean there is no guarantee that that for example right when you would always I mean see the thing is you want to avoid this jump right you want to make the jump shorter so that you do not end up oscillating ok.

There is no guarantee that with this right you will always you will always end up like this but then if you jump you will jump much less for example, you will jump to e and not to d because of the fact that there is already a change in the sign and that sign change will reduce the step size which you eventually take right whereas without that knowledge you would have already made a made a kind of big big step size I mean you understand

intuitively right what this means right. So, it is like saying that if I had known that well my sign had already changed there I mean then maybe I would have actually you know taken a smaller step where would have gone all the way to d right I would have probably reduced my step size right and that is the idea ok. So, so, so, so, so if you actually implement this right people have found that the number of oscillations that you get are get are far fewer because because it actually helps you know right a priori that that you should not have taken a large step and this is mainly mainly useful when you are actually near near a local minimum the rest of the places it does not really matter ok. But then when you are near a local minimum it will help you prevent you know undergoing too many oscillations because oscillations then it means you know more iterations and spending more time right getting there you will get there eventually but then you know unless of course you know you make steps like the one that I showed last time that you simply go off the off the cliff right.

Yeah. So, I am not writing the explanation at all, but I hope right you have actually understood what this what this means ok and this sometimes also called NAG ok it is called Nesterov accelerated gradient ok. So, some people call this as NAG which is Nesterov accelerated gradient accelerated gradient this is some 83 work. Then the next thing right that that we should that I thought we should look at this this alpha guy itself. So, as far as the momentum is concerned right these are the two main things that that one should be aware about.