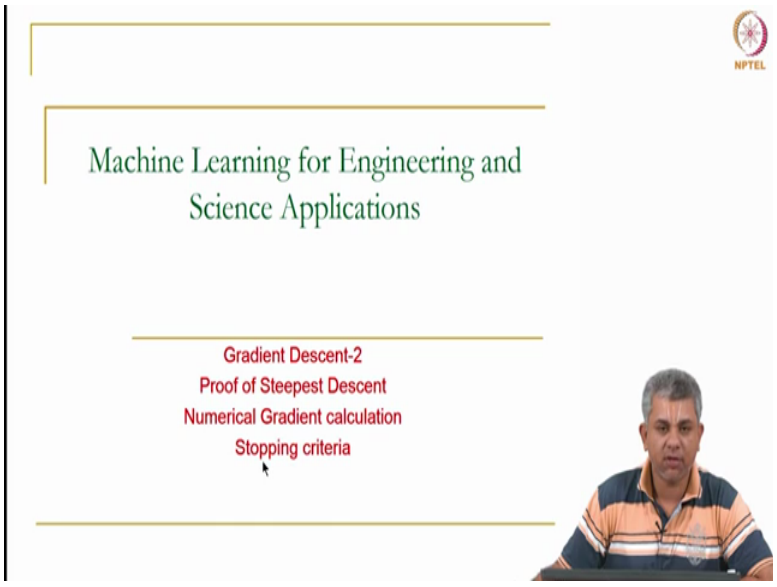
Machine Learning for Engineering and Science Applications

Professor Dr. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology, Madras

Gradient Descent – 2 Proof of Steepest Descent Numerical Gradient Calculation Stopping Criteria

(Refer Slide Time: 0:15)

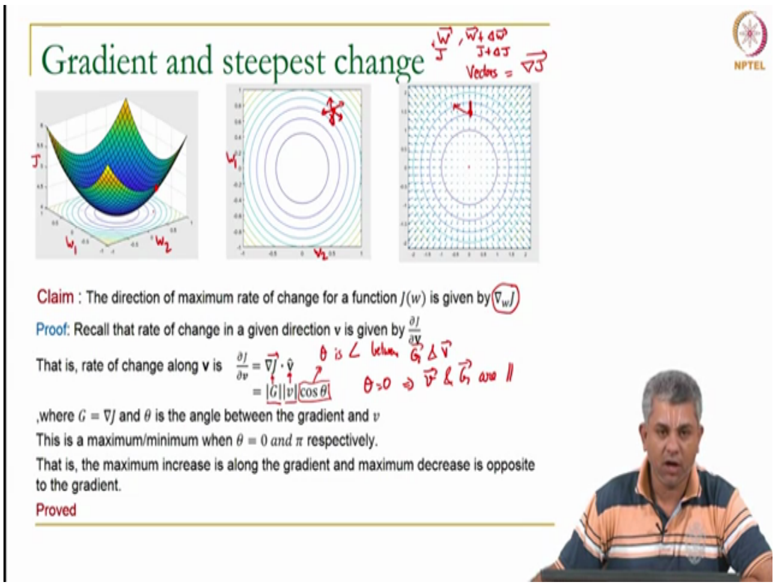


Machine Learning for Engineering and Science Applications

Gradient Descent-2
Proof of Steepest Descent
Numerical Gradient calculation
Stopping criteria

In this video we will be looking at some of the details of gradient descent which we skipped in the last video. At first we will be looking at proof of the steepest descent property, we will also look at how to calculate numerical gradients in case an analytical expression for J is not available and finally we will look at when to stop your iteration or atleast some criteria which people use commonly for this.

(Refer Slide Time: 0:40)



Gradient and steepest change

Claim : The direction of maximum rate of change for a function $J(w)$ is given by $\boxed{\nabla_w J}$

Proof: Recall that rate of change in a given direction v is given by $\frac{\partial J}{\partial \mathbf{v}}$

That is, rate of change along v is $\frac{\partial J}{\partial v} = \vec{\nabla J} \cdot \hat{v}$

$= |G||v|\boxed{\cos\theta}$

θ is $\angle$ betw $\vec{G}$ & $\vec{v}$

$\theta = 0 \Rightarrow \vec{v}$ & $\vec{G}$ are $\parallel$

,where $G = \nabla J$ and θ is the angle between the gradient and v

This is a maximum/minimum when $\theta = 0$ *and* π respectively.

That is, the maximum increase is along the gradient and maximum decrease is opposite to the gradient.

Proved

Okay, so once again let us look at the kind of functions that we were looking at before with this being the landscape and this being the contour of the function remember this is J and let us arbitrarily name this w 1, w 2, okay. So if you are at some point let us say here some w 1, w 2. You have some value of J. So remember that these lines represent values of J, similarly here you could look at it in 3D also there is some value of J.

Now you have a several choices from here, you could move in any direction, whichever direction you move in so if you have if you are at some w and you move to w plus delta w vector instead of having J you will go to some J plus delta J. So the question is what gives you which direction gives you the maximum average, in which direction will the rate of change of w be the maximum?

Now what I have plotted here are actually vectors which are equal to gradient of J, so you can see that as they come near the origin they become smaller and smaller and at every point grad J itself points in a certain direction it is a vector it has a magnitude, it also has a direction. So claim of the steepest descent theorem is some sense is that the direction of maximum rate of change for a function J of w is given by grad J, so if you take a unit vector in that direction and you move in that direction that is the direction in which (temperature will change oh not temperature) sorry J will change the maximum.

So just to give you an example if you are in a room and there is a heater at one end and there is a air conditioner at another end if you join the heater to the air conditioner in one straight line so that is where your temperature will change the most rapidly. In all other directions the temperature will change but it will change in a slightly different speed. So the direction in which this change is maximum turns out to be the gradient let us proof that very shortly.

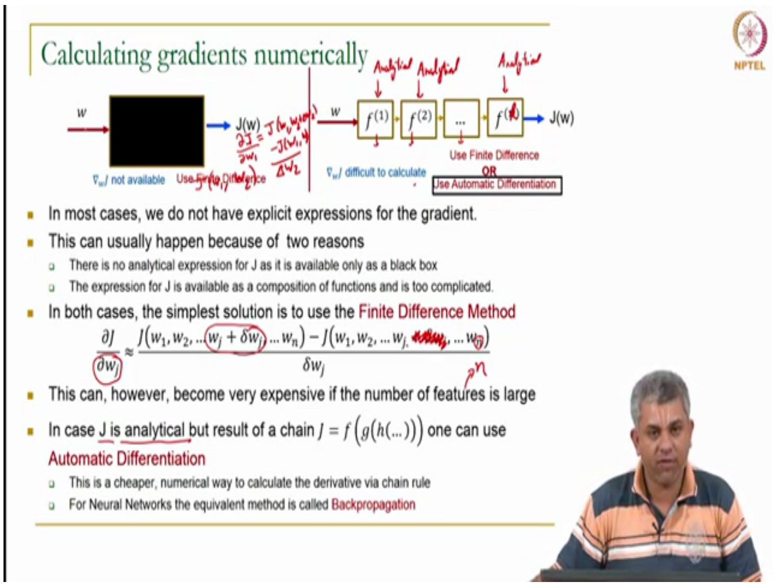
So remember that if you want rate of change of a scalar which is J, remember we are always dealing with scalar cost functions in the scores, if you want its rate of change in a given direction that is given by del J del V vector assume for now that V vector is a unit vector, also remember this also we saw that del J del V vector can be written as grad J dotted with V cap, so this is what will give you del J del V vector.

Now since this is a dot product you can write this as mod of G where G is nothing but grad J vector multiplied by value of norm of V or absolute value of V or length of V multiplied by Cosine theta where theta is the angle between G vector and V vector this is the simple definition of dot product. So G is grad J and theta is the angle between the gradient and V. Now since this is the value of change of J in any direction we can get a maximum remember norm G is fixed, norm V is also fixed, what is variable is the angle that you give between the two.

So when is this going to be a maximum? This is going to be maximum when theta is 0, and when is this going to be minimum? It is going to be minimum when theta is Pi. So this is pretty straight forward. So when is theta 0? Theta is equal to 0 means V and G are parallel, in other words when you choose a direction that is along the gradient you will increase most rapidly and when you choose a direction which is directly opposite to the gradient which is exactly 180 degrees away from the gradient you are going to decrease most rapidly.

You can see this in this figure also, you see the gradient is actually this is another property we are not discussing it, it is normal to the contours. So if I am here and if I want to increase most rapidly, the next value is here, the best direction to go in is directly perpendicular. If I go here I will have to go further in order to get the same change. So (that is what this property choses) that is what this property represents, what it says is the rate of change is maximum right along the gradient and the rate of change is minimum or maximum negative in the opposite direction, so this is the proof of this theorem.

(Refer Slide Time: 5:44)



Calculating gradients numerically

Analytical
Analytical
Analytical
w
J(w)
$\frac{\partial J}{\partial w_1} = \frac{J(w_1, w_2 + \delta w_2) - J(w_1, w_2)}{\delta w_2}$
$\nabla_w J$ not available
Use Finite Difference
w
$f^{(1)}$
$f^{(2)}$
...
$f^{(n)}$
J(w)
$\nabla_w J$ difficult to calculate
Use Finite Difference
OR
Use Automatic Differentiation

- In most cases, we do not have explicit expressions for the gradient.
- This can usually happen because of two reasons
  - There is no analytical expression for J as it is available only as a black box
  - The expression for J is available as a composition of functions and is too complicated.
- In both cases, the simplest solution is to use the Finite Difference Method

$$\frac{\partial J}{\partial w_j} \approx \frac{J(w_1, w_2, \ldots (w_j + \delta w_j) \ldots w_n) - J(w_1, w_2, \ldots w_j, \ldots w_n)}{\delta w_j}$$

- This can, however, become very expensive if the number of features is large
- In case J is analytical but result of a chain $J = f\left(g\left(h(\ldots)\right)\right)$ one can use Automatic Differentiation
  - This is a cheaper, numerical way to calculate the derivative via chain rule
  - For Neural Networks the equivalent method is called Backpropagation

Next we wanted to find out how to calculate gradients numerically? Remember as I said earlier we do not have explicit expressions for the gradient available in most cases, this can happen due to one or two reasons, there is actually genuinely no analytical expression at all for J and it is available only as a black box so this is one case so grad J is not available. Another case where this can happen is J is actually available as a composition of functions.

An example of that is J is so w runs through a box f 1, it runs to another box f 2, f 3, f 4, up till so on and so forth an f n and even though you have analytical expressions for each one of these J is actually very hard to calculate because it is a very very lengthy expression. In both cases a very very simple solution that you can use is actually you something called the finite difference method, what is a finite difference method? It is simply our definition of derivative written down a little bit more explicitly.

So suppose you have J as a function of w 1 and w 2 and you want del J del w 1 you would say del J del w 1 is w 1 w 2 plus delta w 2 minus J of w 1 w 2 divided by delta w 2, in the case of w which has n components or what is called n features you simply perturb the particular derivative you are interested in, so if you are interested in the third derivative you perturb the third variable so this should actually not be there this is an error this should be w 2 w J here and w n divided by delta w J so you have to perturb each of the variables.

So you can use finite difference in both cases. However, if your number of features or number of features simply is n which is the size of your w, if your number of features is very very high this can actually become very expensive because you will have to perturb each one, so if you have let us say 500 features for example as we were discussing earlier if you have a 60 cross 60 image and you have like 3600 you will have to perturb it 3600 times in order to find out each one of these derivatives so this can become very very large.

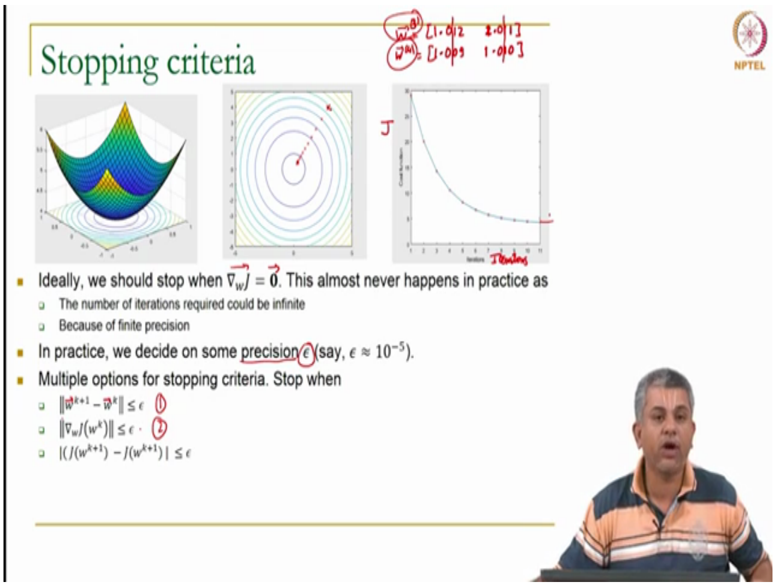
A different method which is what has made neural networks practical of late is what is called automatic differentiation. So this is very useful in case you have an analytical expression but the analytical expression is hidden as a chain. So suppose you have analytical expression for f 1, f 2, up till f n instead of f n we can call it k so that you do not get confused with this n.

So let us say you have k such chains and k being 100 which is very common and something like deep learning. So in each one of these cases let us say you have this expression as a linear expression, this is quadratic, this is linear again it is impossible to write it down and differentiate it by hand atleast, which is why the computer does the analytical differentiation and this is called automatic differentiation.

You use either finite difference or use automatic differentiation, and automatic differentiation is the method of choice in case this is what is happening which is you have analytical expression but it is a result of multiple chains. So we will see this within the context of neural networks this is called back propogation when we do this for neural networks and that is what has made modern neural networks possible ever since the 70's or 80's when the algorithm was first formulated.

So just as a summary in case you really have no other choice and J is only numerical then you use finite difference but in case J is available as a chain of functions for example f of g of h of w of something, if you have something of that sort then we use automatic differentiation.

(Refer Slide Time: 10:14)



Stopping criteria

$\vec{w}^{(3)} = [1.012 \quad 1.011]$
$\vec{w}^{(4)} = [1.009 \quad 1.010]$

- Ideally, we should stop when $\vec{\nabla_w J} = \vec{0}$. This almost never happens in practice as
  - The number of iterations required could be infinite
  - Because of finite precision
- In practice, we decide on some precision ϵ (say, $\epsilon \approx 10^{-5}$).
- Multiple options for stopping criteria. Stop when
  - $\|\vec{w}^{k+1} - \vec{w}^k\| \leq \epsilon$ ①
  - $\|\nabla_w J(w^k)\| \leq \epsilon$ ②
  - $|(J(w^{k+1}) - J(w^{k+1})| \leq \epsilon$

The final topic for this video is what is called stopping criteria. So ideally we should stop our iteration, we are trying to find out optimal w, ideally we should stop it when grad J actually becomes 0 remember 0 means not just 0 value 0 vector. This almost never happens in practise because as we saw with the alpha equal to 0.1 case your number of iterations could be actually infinite, also you cannot give 0 to infinite precision, your machine itself has finite precision. So because of these two reasons you actually want to stop quite early or a little bit earlier.

So once again let us look at our previous example we started somewhere and we wanted to go till the origin instead of using infinite precision so chose some finite precision. I will say I am happy with five decimal places in which case I will set some precision called epsilon I will call it standard notation is epsilon to say that I am happy with 10 power minus 5 or five decimal places of accuracy, but five decimal places of accuracy in what?

So we have multiple options, one thing is our friend the norm returns here, remember w is a vector so if I am doing iterations so if you recall the iterations we did there suppose some iteration w vector is 1.012 and 1.011 this is w in this third iteration let us say and let us say in the fourth iteration this is 1.009 and 1.010 so up till two decimal places we can be fairly certain that we have got the answer reasonably correct, specially this repeats again and again.

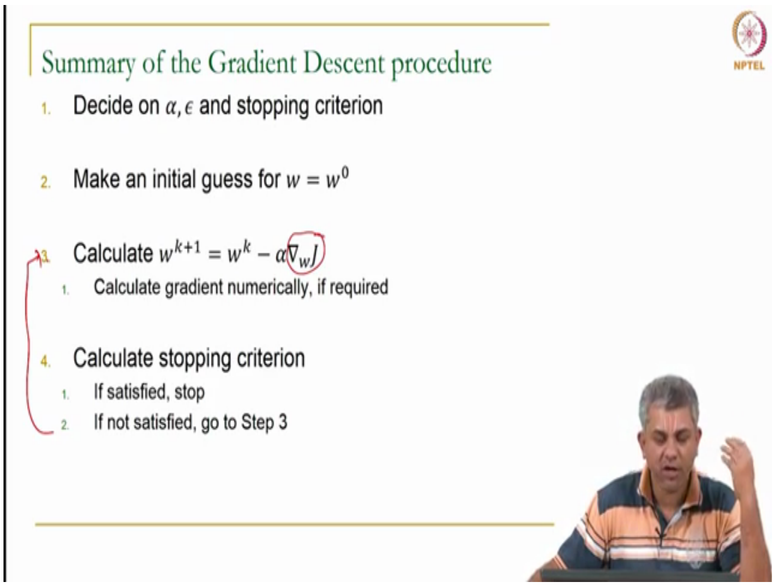
Now remember of course this is a vector, this is a vector how do we find out one vector is how closed to the other vector? By subtracting them and taking the norm. So if you subtract them and take the norm typically any norm it could be 1 norm, 2 norm infinity norm that is your choice, it will come out to be a number and I say that the difference between the previous vector and the next vector should be smaller than the given precision and this precision could be in this case the example I have given is 10 power minus 5.

Now instead of saying w has to be close to each other I could also say something like grad J should be smaller than a given number, remember we were trying to do grad J equal to 0 and instead of saying grad J equal to 0 I could say the moment grad J becomes lower than 10 power minus 5 I will stop. So some iteration of this sort could happen so you could start with some w and you could end up at some other w which is closer and closer to each other.

Finally, instead of looking at grad J, so this is choice 1, this is choice 2, I could also look at the cost function itself which is the J for the previous value remember our example we were going from 29 to 229, etc when it was diverging but it was slowly coming to 4 when it was converging. So I could look at J of the previous cost of the previous iteration and look at cost of the next iteration and find out the difference between the two.

So here is a standard figure that is often drawn, this is J, this is number of iterations and people often visually track the convergence you cannot track the convergence of w because w is lords of vectors, but you can track the convergence of the cost function. So here it started at some value, in our case it started with 29 and slowly started decreasing and you can get satisfied that may be after sometime it will kind of converge some value and you could stop here. So we can look at the difference in J's so this is the third choice that we have.

(Refer Slide Time: 14:16)



Summary of the Gradient Descent procedure

1. Decide on α, ϵ and stopping criterion
2. Make an initial guess for $w = w^0$
3. Calculate $w^{k+1} = w^k - \alpha \nabla_w J$
   1. Calculate gradient numerically, if required
4. Calculate stopping criterion
   1. If satisfied, stop
   2. If not satisfied, go to Step 3

So here is a quick summary of the gradient descent procedure, you first have to decide on the hyper parameter learning rate alpha, you can also decide on epsilon and you can also decide on what stopping criteria you will use, will you use it based on w, based on grad J or based on J itself, make an initial guess for w, calculate the next guess, if required you will always have to calculate the gradient find out either the gradient numerically or analytically whichever way it is available to you, calculate the stopping criteria.

If suppose the stopping criteria is satisfied let us say J, the difference between J now and J before is less than the epsilon that you decided then you stop, if not repeat the iteration. So this is summary of the gradient descent procedure and this procedure will be used with small variants with minor variants throughout this course especially throughout the deep learning module, thank you.