2:34:07

Module 5

Gradient Descent.

So we have looked at the NR method that uses the gradient to iteratively solve a 1D function of x, say.

Now, we will generalize that to figure out how to do something similar with a multidimensional function, function with multiple variables, and how to use the gradient to find the maxima and minima of such a function

Later on, we will let us optimize, the best fit, for the parameters of function we trying to fit and

Say I have the following function: x^2y

$$f(x,y) = x^2y$$

What we have here is a function that gets big and positive, when x is big, and y is positive and gets negative, when y is negative

So if I look down the y axis, I get a projection of straight line

And if I spin to look down the x axis, I get an upright parabola for y positive, and one going down for y negative

and function is equal to zero, along both the axis

Now at bottom here in matlab, I have used the surf function to plot contours where the function has a constant value, just like the contours of constant height on a topographical map

So question here, how do I find the fastest or steepest way to get down this graph.

Previously we found out what the gradient of function is wrt each of its axis

So you can find the $\frac{df}{dx}$ just by differentiating the function f, treating all the other variables as constants

So in this case we have: function f of x, y

$$f(x,y) = x^2y$$

$$\frac{df}{dx} = 2xy$$

$$\frac{df}{dy} = x^2$$

Now, we can write down these 2 gradients in vector which we call the "grad" of f, ∇f

$$\therefore \nabla f = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}$$

writing down df/dx, df/dy in x and y location of vector.

Grad is like the most awesome vector ever.

Grad is little vector, that combines LA and Calculus together

Let's say:

$$\nabla f = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \cdot \begin{bmatrix} a \\ 0 \end{bmatrix}$$

we take vector, and we dotted that, with grad f

(⇒ we moving df/dx along x axis
⇒ some amounts of dx/dy along y axis)

The important part, it's evaluated at (a location)

$$\therefore \begin{bmatrix} \frac{df}{dx}(a,b) \\ \frac{df}{dy}(a,b) \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix} \hat{r}$$

at f has some value at space (a,b)

∴ evaluated at a location.

Q: if we want to know how much the function will change when we move along some unit vector in arbitrary direction, let's call that our vector $\hat{r}$, with components c and d

$\therefore df = \frac{\partial f}{\partial x} c + \frac{\partial f}{\partial y} d$

- this is called the directional gradient

Another question: what is the maximum value this directional gradient can take

where ∇f is a vector, and dotting it with $\hat{r}$ and θ's: what is the maximum value, given $\hat{r}$ is a unit vector

- nearly hang left (θ), when we dot them together is the $\cos\theta$ term, because it's $ab\cos\theta$
- and that will be maximized when $\cos\theta = 1$
- which means $\theta = 0$, which means $\hat{r}$ is parallel to ∇f

So in order to find the maximum value of the directional gradient, we want $\hat{u}$ that's actually the normalized version of ∇f (grad)

So we can write it down (do not calculate):

$$\hat{u} = \frac{\nabla f}{\|\nabla f\|}$$

$\therefore$ $\hat{u}$ dotted with the normalized version of itself

$\therefore$ $\hat{u}$ divided by its modulus

But $\hat{u} \cdot \nabla f$ dotted with itself is just equal to (the square of $\|\nabla f\|^2$)

$$\nabla f \cdot \frac{\nabla f}{\|\nabla f\|} = \frac{\|\nabla f\|^2}{\|\nabla f\|} = \|\nabla f\|$$

and that's the steepest gradient we can possibly have

Other Question: which way does "grad" point

Not as easy to see, but grad points up to
the direction of steepest descent, perpendicular
to Contour lines

So, if you on a mountain and it's foggy and
you can only see locally around you, and
you want to go down hill.
You don't walk around the Contour, you go
down hill

So if Contours to my side, and I stare down
hill, then down the hill is 90° to Contour line

Think about ways to find, if it's up or down,
df/dx is positive if you going up, so actually
grad f ~~of~~ points up hill, ~~and~~ in steepest way,
and - (minus) grad point down the hill,
~~the~~ steepest way.

So grad points up the direction of steepest descent

So if we have some data science problem, where we want to minimise the difference between our data values and model fit, then what we want to do is find the lowest point in function, the function of badness, we want to find the best, we want badness minimised. to find point where the

"Newton Raphson"

like NR, we can use the gradient to go from some trial point down towards the solution.

But in NR, we trying to find the zero point

Here we do not know what the minimum value of the function is

So we do not know how far down we need to go

So if we somewhere on a mountain

But we do not know the altitude of the value, so what we do, what's called the GRADIENT DESCENT method., we take a series of little steps down the hill

Basically if we start at some position s_n, then our next position s_{n+1} is given by s_n plus one little step down hill, and that little step is given by minus [some amount] times ∇f

$$\therefore s_{n+1} = s_n - \gamma \nabla f(s_n)$$

and grad evaluated at some previous position (s_n)

On graph, that's going to look like taking a little step and reevaluate and make another another s_{n+1}, and make another step and take series of steps down hill If we overshoot, that's ok, since grad will just take us back to minimum.

And notice, as the gradient gets shallower,
as we ~~get~~ come towards the turning point,
then the steps automatically gets smaller,
as grad gets smaller.

This is quite a nice method.
There are lots of ways to enhance it
Its very powerful and simple.
The other things to notice is that multiple
local minima in Landscape, and we
may get stuck in one of them.
Of course which one we find, depends
on our starting point.
We want find the ones, we will find
one at time
So in general case, there are some problems,
but nevertheless
This is quite a neat method, just
of taking little steps down hill.

This is probably the main way (in which most) numerical optimizers work that people use in real world.

Recap:

- We introduced ourselves to 'grad' vector, ∴ gradient vector.
- And merged Calculus and Vectors together to take our first steps in vector Calculus
- This allowed us to find method ~~to just~~ for just using the gradient to step our way forward solving a problem, for multivariable case, which is called GRADIENT DESCENT