

Gradient Descent.

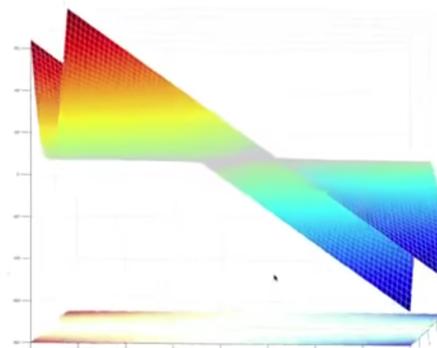
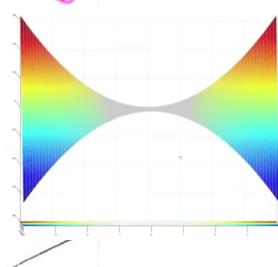
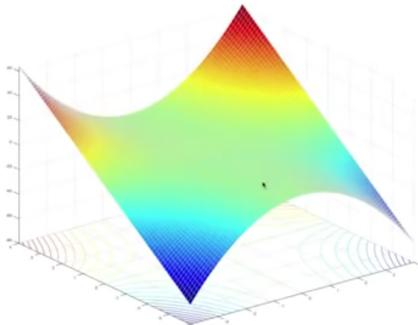
So we have looked at the NR method that uses the gradient to iteratively solve a 1D function of x , say.

Now, we will generalize that to figure out how to do something similar with a multidimensional function, often 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

Say I have the following function:

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



$$x^2y$$

What we have here is often when x gets big and positive, when y is big, and g is positive and gets negative, when y is negative

So if I look down they axis, I get a proportion 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 gradients equal to zero, along both the axes

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

Question here, how do I blend the fastest or steepest way to get down this graph -

Previously we found out what the gradient of functions wrt each of its axes

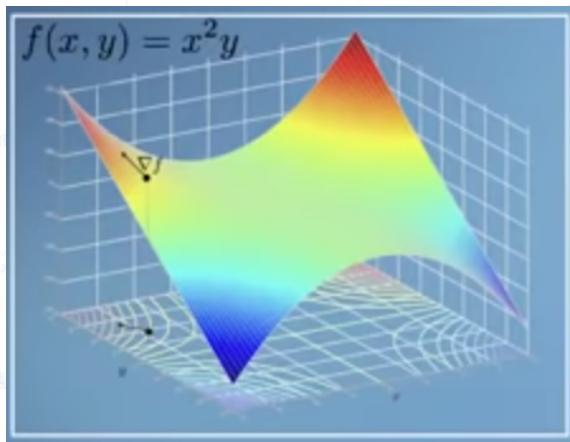
So you can find the $\frac{df}{dx}$ just by differentiating the function f , treating all the other variables as constants.

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 the 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 isolation of vector.

Grad is like the most awesome vector ever.

Grad is little vector, mat combines LA and Calculus together

lets say:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\cdot \begin{bmatrix} dx \\ dy \end{bmatrix}$$

we have vector, and we dotted that, with gradf

(\Rightarrow we moving df/dx along x axis)

\Rightarrow some amounts of dx/dy along y axis)

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

$$\begin{bmatrix} \frac{\partial f}{\partial x} & (a,b) \\ \frac{\partial f}{\partial y} & (0,0) \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix}$$

\hat{r}

at f has same values at space (a,b)

\therefore evaluated at a location.

So if we want to know how much dependent will change when we move along some

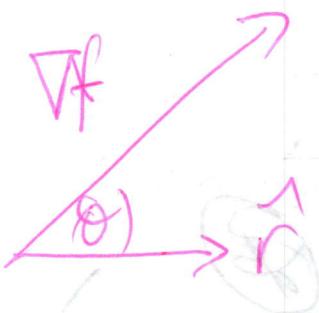
unit vector in arbitrary direction, lets call

that our vector \hat{r} with components add

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

This is called the directional gradient

Another question: what's the maximum value
the directional gradient can take



where ∇f as a vector, and dotting it with \hat{r}

and Q's: what is the maximum value, given

\hat{r} is a unit vector

mainly being left (θ), when we dot them together
is the $\cos\theta$ term, because its $\hat{r} \cdot \nabla f = \|\nabla f\| \cos\theta$

and that will be maximized when $\cos\theta = 1$

which means $\theta = 0$, which means \hat{r} is
parallel to ∇f

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

Or we can write it down (do hat calculation):

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

- ∇f divided with the normalized version of itself
- ∇f divided by its modulus

But $\nabla f \cdot \nabla f$ divided 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 possible have

gradient we can

never interpret
anything not ∇f

Other Question: Which way does "grad" point

8

It's like 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 really around you, and you went 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 always 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 9.

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 is kind of the "Newton Raphson" we went to find the best, want badness, we went to find the best, want to find point where the badness is minimised.

Like NR, we can use the gradient to go from some trial point down toward 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.

Or we do not know how far down we need to go.

Do if we somewhere on a mountain

But we do not know the attitude of the value,
so what we do, what's called the GRADIENT.

Descent method., we take a series
of little steps down the hill.

Rate of descent at same position s_n ,
then our next position s_{n+1} is given by s_n plus
some little step down hill, and that little
step given by $-\frac{\text{grad}}{\|\text{grad}\|}$ times $\sqrt{f(s_n)}$

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

and grad evaluated at same previous position (s_n)

On graph, that's going to look like taking
a little step and reevaluate and make
another step and reevaluate and make another

another step and take series of steps down hill
@ 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 closer towards the turning point,
then the steps automatically get smaller,
as grad gets smaller.

This is quite a nice method.

There are lots of ways to enhance it.
It's very powerful and simple.

The other thing 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 others, we will find
one at time

In general case, there are some problems,
but nevertheless

This is quite a neat method, got
of many 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
- We learned how to find methods to just for
just using the gradient to step our way
forward solving a problem, for multivariable
cases, which is called GRADIENT DESCENT