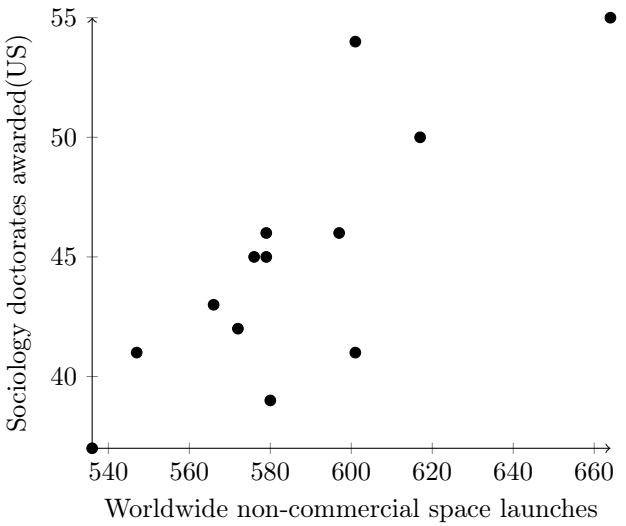
Gradient Descent: The Beginning of the Robot Apocalypse

It's not going to be funny when it's *you* being localized by the killbots

What is it and why do I care?

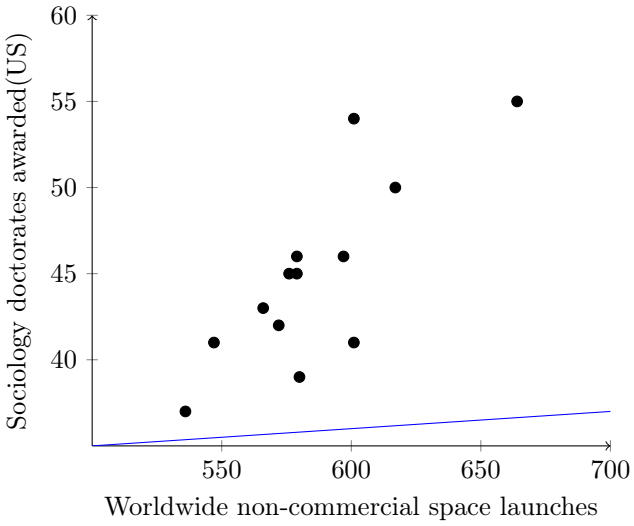
So say we have the following graph:



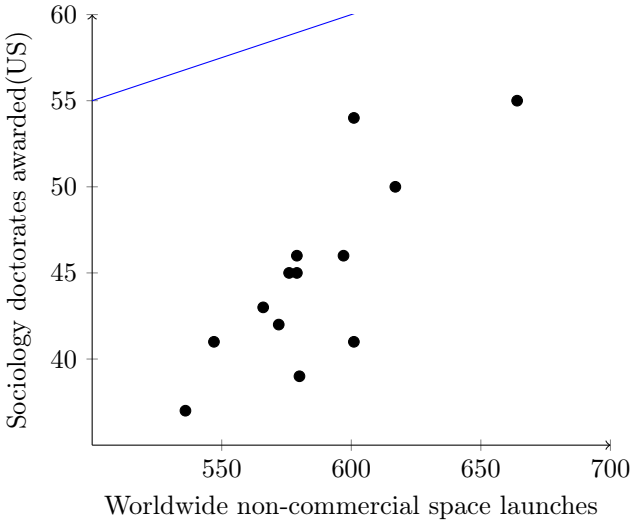
Let's find the line of best fit between the number of worldwide non-commercial space launches and sociology doctorates awarded in the US. How hard can it be?

Ok so let's just think of some values for a simple linear fit, $y = mx + b$.

Maybe... $\boldsymbol{m = 0.001}$, $\boldsymbol{b = 30}$?



Ok, no, that sucked. Let's try $\boldsymbol{m = 0.001, b = 0}$?



Wow, humans sure suck at things. So how should we fit the line? Calculators probably don't just randomly guess... right?

If you guessed "right," you're correct. I think.

It's time for...

Gradient Descent!

So, what do we want to minimize? Well, error.

But how do we define error?

Here's an idea. Let's define it in a completely arbitrary way: mean square error. What this means is that $error_{total} = \frac{\sum error^2}{numofpoints}$, or that the total error is defined as the average of the squares of each of the individual errors, or distance from points to the line.

Thanks to our friends Gibbs and Heaviside, we can find the shortest distance between a point and a line using vector projection, and generalize it to define the distance from any point (x_0, y_0) to a line $ax + by + c$ as

$$error = \frac{|a \cdot x_0 + b \cdot y_0 + c|}{\sqrt{a^2 + b^2}}$$

Of course, b is effectively redundant since a and c is all you need to fully define a 2D line, so we can just set b to 1. So, with that being said, let's transform our above equation, knowing this, into slope-intercept form.

$$error = \frac{|-a \cdot x_0 + y_0 + -b|}{\sqrt{(-a)^2 + 1}}$$

I know this is true because I graphed it in desmos and it looked like it worked.

Anyways, our graph of worldwide non-commercial space launches vs. sociology doctorates awared in the US has a total of 13 data points. Meaning this is going to kind of suck.

Anyways, here's the entire equation defining the error function of our line given the paramters m and b. You're just going to have to take my word on the data because tables are hard apparently.

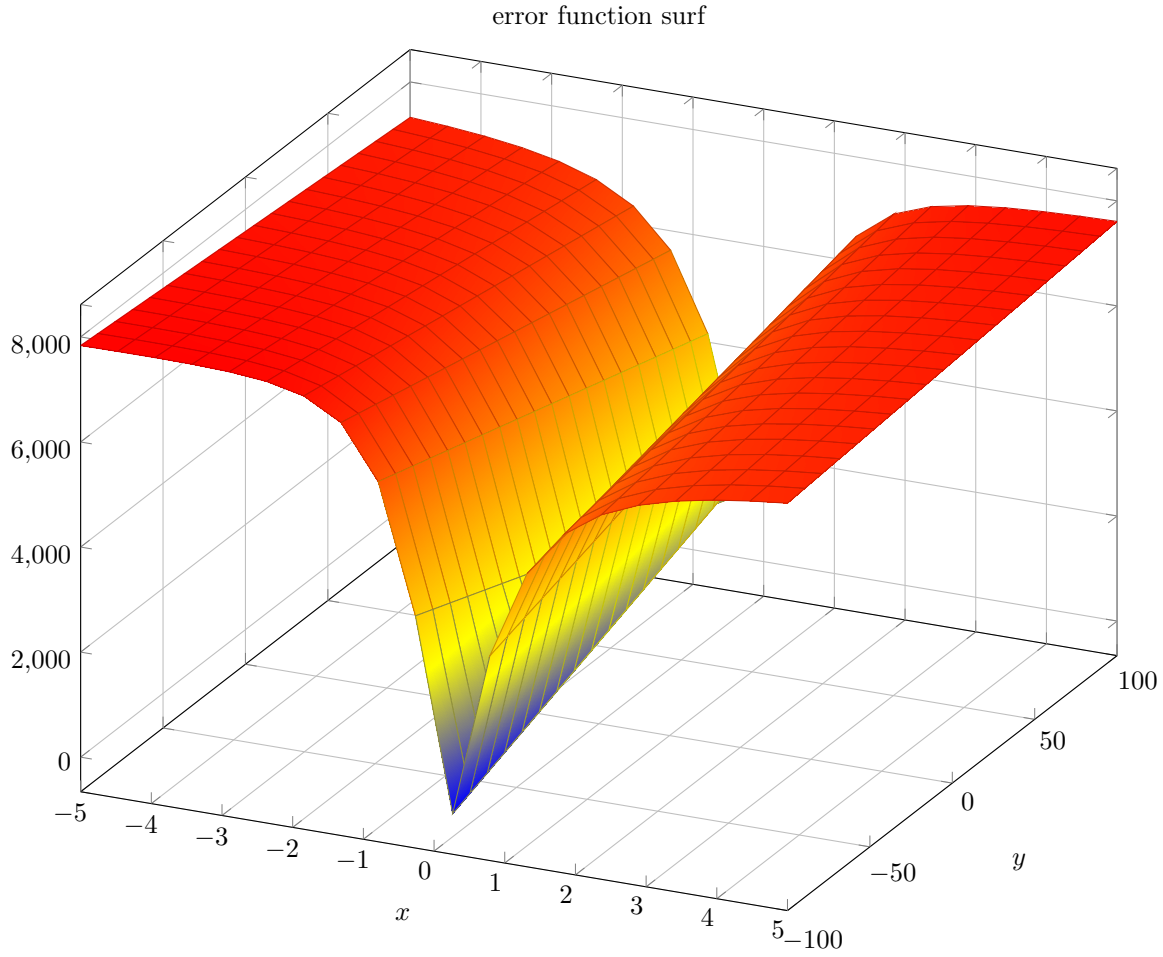
$$\frac{|-m \cdot 601 + 54 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 579 + 46 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 572 + 42 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 617 + 50 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 566 + 43 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 547 + 41 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 597 + 46 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 580 + 39 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 536 + 37 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 579 + 45 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 576 + 45 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 601 + 41 + -b|}{\sqrt{(-m)^2 + 1}} + \frac{|-m \cdot 664 + 55 + -b|}{\sqrt{(-m)^2 + 1}}$$

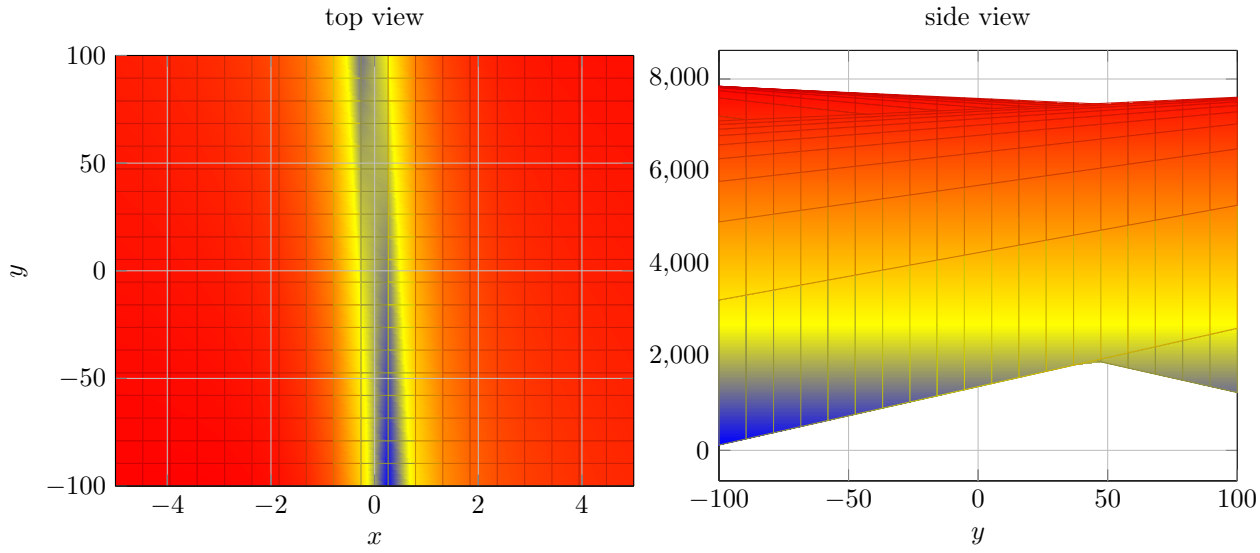
Luckly, assuming m is positive, this all compresses down to:

$$error = \frac{|584 - 7615x - 13y|}{\sqrt{m^2 + 1}}$$

This is cool. We now have an equation giving us an *error function* of our parameters to the linear equation (m and b).

So what does it look like if we graph this?





Wow thats cool, it's a really bad paper plane. Yay now we have the error function, and this paper is over. Cool see ya thanks.

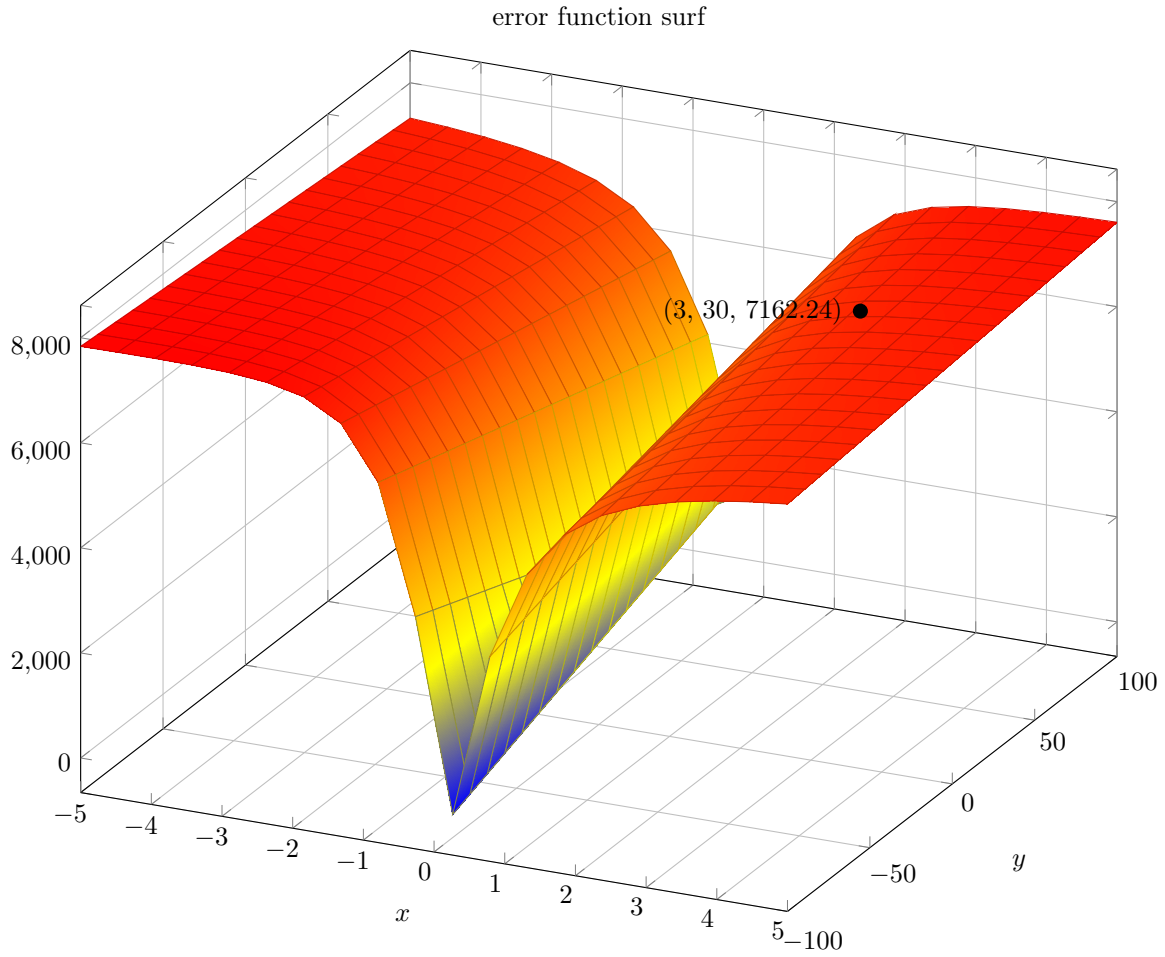
but wait there's more

ok but so how do we get the line of best fit from this?

Using ~~***SCIENCE!***~~ ***MATHEMATICS!***

So let's pick a point (x, y) on the domain of $x \in [-5, 5]$ and $y \in [-100, 100]$.

How about $(3, 30)$? Inputting that into our error function gives ≈ 7162.87. That's a lot (I think) but we can minimize that in a second.



So if you recall, the equation for total error is $\frac{|584 - 7615x - 13y|}{\sqrt{x^2+1}}$. So how do we minimize the error based on that? Well if you're in some lame Calc AB course the answer will be find all the minima. Unfortunately, you can't feasibly do that in more advanced equations like differential equations and big neural networks. However, derivatives can still help.

What happens if we take the 2D derivative AKA gradient vector? That's $\nabla f(x_0, y_0)$, or in more specific terms, $\left[\frac{\partial x}{\partial z}, \frac{\partial y}{\partial z}\right]$. Cool. So, entering the total error equation into totally not wolframalpha, we determine the overall vector gradient function to be

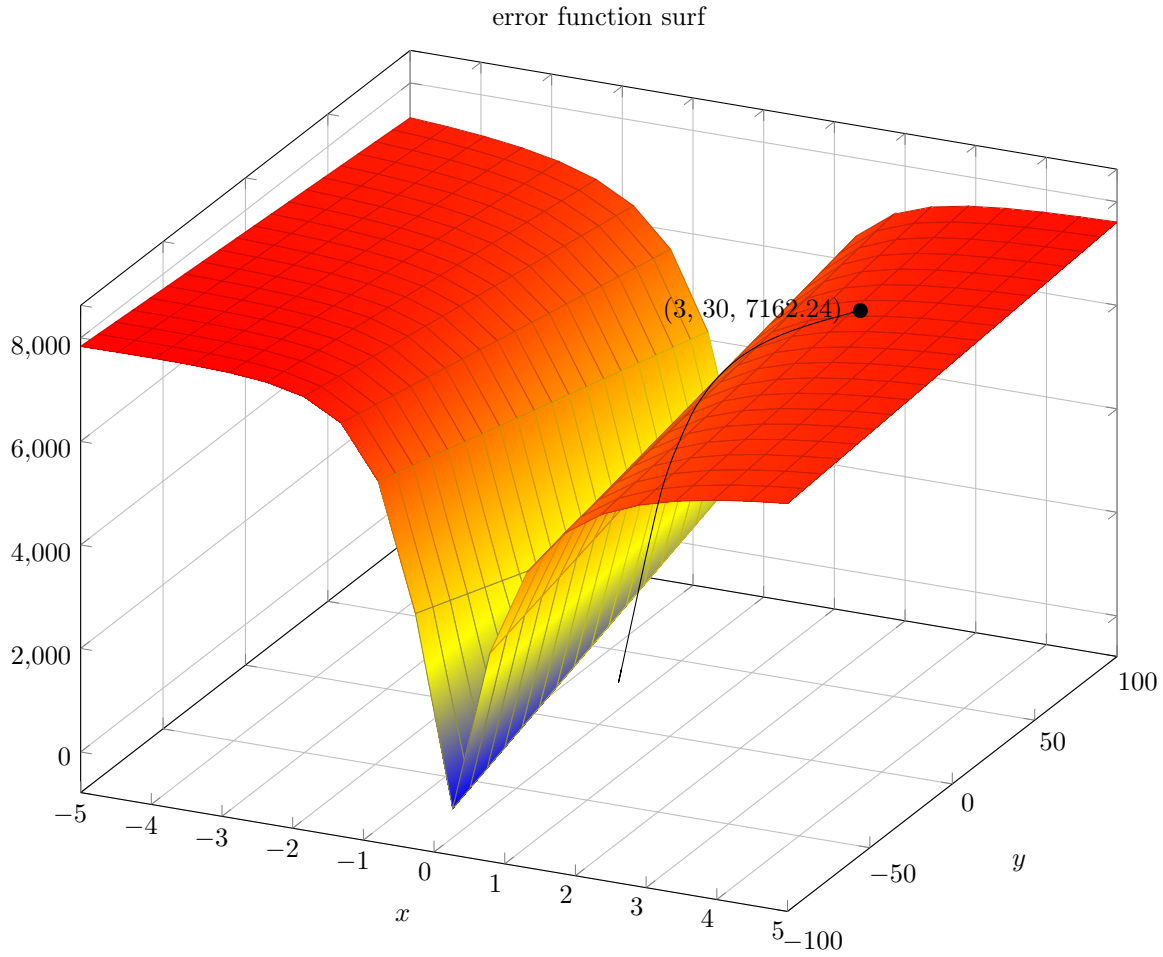
$$\nabla f(x, y) =$$
$$\left[\frac{7615(x^2+1)(7615x + 13y - 584) - x|-7615x - 13y + 584|^2}{(x^2+1)(3/2)|-7615x - 13y + 584|}, \frac{13(7615x + 13y + 584)}{\sqrt{x^2+1}|-7615x - 13y + 584|}\right]$$

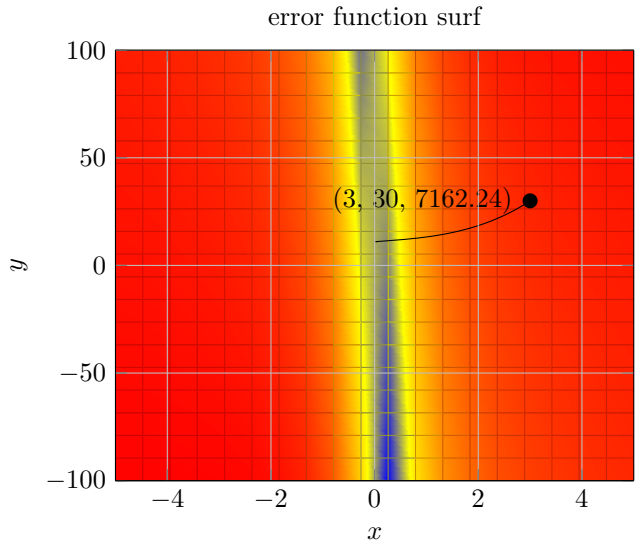
That's a handful, but thankfully we have computers.

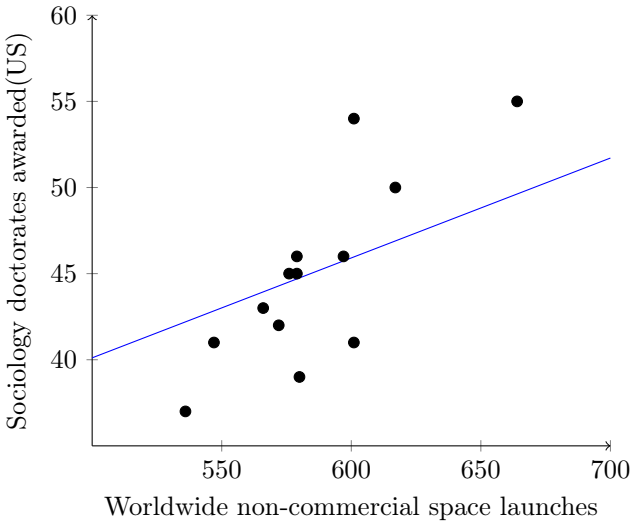
So at $(3, 30)$, that'll be $\left[-260.40, -4.11\right]$. That means that if we want to *descend* along the gradient, we just "step" *away* from the gradient vector (because the gradient vector is the direction of maximum gradient increase).

Let's try that. Keep in mind the current error is ≈ 7162.24 at $(3, 30)$ with a gradient of $\left[-259.40, -4.11\right]$. A step size of $1 \cdot 10^{-3}$ means that we'll update the x-coordinate from 3 to $3 + (-1) \cdot -259.40 \cdot 1 \cdot 10^{-3}$, and the y-coordinate from 30 to $30 + (-1) \cdot -4.11 \cdot 1 \cdot 10^{-3}$. This results in a new point of $\left[-2.7406, 30.00411\right]$. Now, the new error is ≈ 7086.45.

So what if we use the magic power of computers to do this a bunch of times?


error function surf
(3, 30, 7162.24)
8,000
6,000
4,000
2,000
0
−5
−4
−3
−2
−1
0
1
2
3
4
5
x
100
50
0
−50
−100
y


error function surf
(3, 30, 7162.24)
100
50
0
−50
−100
y
−4
−2
0
2
4
x



Et voila, that's gradient descent.

Now normally you'd use a "real" descent algorithm like Stochastic Gradient Descent or Adadelta, not a cobbled-together python implimentation that I'm pretty sure I defined the loss function wrong for. But yeah, that's a kinda cool probably extremely flawed implementation, and the ∇ symbol looks pretty cool.