Stability

Tuesday, August 25, 2020 10:26 PM

... previously, we discussed whether a "problem" is well-conditioned

we abstracted this to $f(x)$ but it could be as complex as predicting the # of hurricanes next year (and x = current weather observations)

now, discuss stability of an algorithm.

To clarify, let our problem be evaluating $f(x) = (x-c)^2$

If this is ill-conditioned then no matter how clever we are at writing code, we may lose a lot of precision. So conditioning is a prerequisite for accurate computations

ie. small condition #

But, there are many ways to implement f, ie., different algorithms

Algorithm 1: temp = $x-c$
output = temp^2

Algorithm 2: output = $x^2 - 2x\cdot c + c^2$

To summarize:

Problem (ie, overall goal) can be well-conditioned or ill-conditioned

Algorithm (i.e., implementations) can be stable or unstable

So what does stable mean?

In some contexts (PDEs, linear operators) we have precise definitions
For now, use the book's vague definition

[An algorithm is stable if] small changes in the initial data produce correspondingly small changes in the final results.

i.e., errors don't compound over time

Another (vague) definition:

An algorithm is unstable if it produces more error than would be expected by the amount of ill-conditioning

i.e., if the relative condition # κ is 10^7, this means we expect to lose about 7 digits of precision.
If our algorithm loses 10 digits, it's an unstable algo.

How do we know if an algorithm is stable or not ?

(1) analyze the conditioning of the overall problem

(2) For the algorithm, check the conditioning of every step, and if any of these exceeds the conditioning of the problem, then the algo is unstable

Ex $f(x) = (x-c)^2$

(1) $\kappa_f(x) := \left| \frac{x}{f(x)} \cdot f'(x) \right| = \left| \frac{x}{(x-c)^2} \cdot 2(x-c) \right| = \left| \frac{2x}{x-c} \right|$

This is somewhat ill-conditioned, depending on x
i.e., $c > 0$ and $|x-c|$ small $\Rightarrow$ ill-conditioned
($x = 1$, $c = 1 + 10^{-5}$, then $\kappa_f(x) = 2 \cdot 10^5$)

(2) Algo 1 : 1. temp = $x - c$
2. output = temp^2

Step 1 : think of as $g(x) = x - c$, so $\kappa_g(x) = \left| \frac{x}{x-c} \cdot \underbrace{\frac{d}{dx}(x-c)}_{=1} \right| = \left| \frac{x}{x-c} \right|$

this isn't good, but it's no worse than $\kappa_f(x)$

Step 2 : think of as $h(x) = x^2$, so $\kappa_h(x) = \left| \frac{x}{x^2} \cdot (2x) \right| = 2$

this is fine.

Conclusion: algorithm is stable

Algo 2 1. temp = $-2xc$

2. output = $x^2 + \text{temp} + c^2$

$\frac{x}{-2xc} \cdot \frac{d}{dx}(-2xc) = \frac{-x \cdot 2c}{-2xc} = 1$

Step 1 : this is fine (condition # is 1)

Step 2 : think of as $h(x) = x^2 + \text{temp} + c^2$

$$\kappa_h(x) = \left| \frac{x}{\underbrace{x^2 + \text{temp} + c^2}_{f(x) = (x-c)^2}} \cdot \frac{d}{dx}\left(x^2 + \text{temp} + c^2\right) \right| = \left| \frac{x}{(x-c)^2} \cdot 2x \right| = \frac{2x^2}{(x-c)^2}$$

this is bad! Algorithm is unstable because the condition # at this step is larger than for the overall problem

i.e., $\frac{2x^2}{(x-c)^2} \approx \kappa_f(x)^2$, $\kappa_f(x) = \left|\frac{2x}{x-c}\right|$

Because we can (design) choose the algorithm, we can try to design **stable algorithms**

So this is a central theme of this class

(in contrast, if your problem is **ill-conditioned**, there's not much you can do about it)

Backward Error

A useful perspective

Suppose we want to find $y = f(x)$ (still could represent something complicated (and often multi-dimensional))

but we do it with our imperfect algorithm $\tilde{f}$

(for now, assume perfect input x)

Then we make an error, i.e., if $\tilde{y} = \tilde{f}(x)$

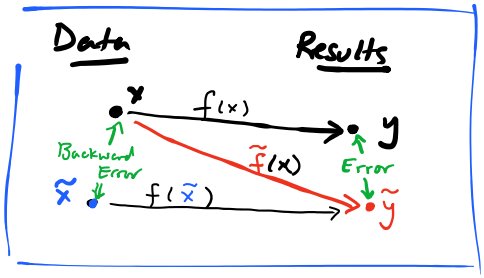
then $\tilde{y} \neq y$ (and $|\tilde{y} - y|$ is the "error")

> Recall, for conditioning, we assumed a perfect algorithm but imperfect input data ... so exactly the opposite scenario

But, suppose we can find $\tilde{x}$ such that $\tilde{y} = f(\tilde{x})$

i.e., we have the right answer ($\tilde{y}$) to the wrong question/problem ($\tilde{x}$)

Then we say $|\tilde{x} - x|$ is the "backward error"



Turns out, if an algorithm always produces small backwards error, then it is stable. (but not vice-versa)

Further Reading: ch 1 in Driscoll & Braun (SIAM 2018)