All posts sorted by recent activity

PSET 3: Assertion Errors in Contour Plot For PS ...
sandipan_dey
3 1d

step size times J-prime No preview available
m_powers
3 2d

Contour plot and f values Hi all, I have a question ...
Wemans
1 4 1d

Computational complexity Hello, What is differen ...
dna47a
4 2w

[staff] Submission not graded? Hello, I have gott ...
HusamMalkawi
18 2w

Stability of Implicit Methods No preview available
sandipan_dey
6 3w

PSET2 plot_Y return values: grader accepts axes ...
sandipan_dey
2 2w

FE code submissions The grader passed my init ...
m_powers
9 4w

Parachute is getting deployed much earlier than ...
sandipan_dey

step size times J-prime

m_powers 2d

I'm talking through my intuitive understanding of the algorithm here, so please excuse or correct any imprecisions in my terminology.

We are trying to find some location which I'll call a^* along the "x-axis" at which the function $J(a)$ is at a minimum. To do this, we are nudging successive guesses of a^k along the x-axis in some direction determined by the sign of J'. So far, so good.

My question relates to how *far* we nudge a^k in the given direction. In particular, why is it useful to multiply our nudge by the magnitude of J' and not just its sign?

Asked another way: sometimes J', which represents a "slope", is high, let's say 10, meaning that J at that point is rising 10 units for every 1 unit of "run". Other times, it might be low, let's say 0.1, in which case J is "running" 10 units for every 1 unit of rise. Why is it useful to nudge a^k 100 times farther along the x-axis in in the first example vs the second?

EDIT: thinking through a little more, ultimately I suppose we are looking for a place where J' is 0. So if J' at a given a^k is low, it means we are "getting close" and so we need to take smaller nudges to zero in without overshooting. But additional discussion is welcomed ;)

Related to 13.1 Single-variable Optimization / 13.1.4 Gradient descent algorithm in one dimension

Showing 2 responses Newest first

darmofal Staff 2h

Good question. I agree with sandipan_dey that in 1D, just heading in the right direction will work. However, when you get close to the minimum, you could easily take too big of a step if you are only using the sign of J' (which is why having the step be proportional to J' is useful, since it naturally decreases the step size as you get close to $J' = 0$).

In 2D, I am pretty sure you can do a similar algorithm in which you take a step so that a_i changes opposite the sign of $\partial J / \partial a_i$ without using the actual value of that derivative. And that this will also find a minimum (though again you will need to be careful as you get close to $\nabla J = 0$. However, the most rapid decrease in J occurs in the direction of $-\nabla J$.

sandipan_dey 1d

J' (or the vector ∇J when generalized to multiple dimensions) in general represents a direction and not just sign.

For example in 2D the direction $[-1, 1]$ is different from let's say $\nabla J = [-3, 2]$, so considering signs only is not enough, if we want to reach the local optimum following the gradient direction (of steepest change in J).

Although in 1D it's just scaling and GD would work even with sign instead of magnitude (resulting in slower convergence or adjustment in learning rate α would be needed to obtain convergence), but for multi-dimensional case it may lead to a completely wrong direction following which we may not even get to the local minimum.

Add a response