Gmail **Nikolas Wolfe <nikwolfe7@gmail.com>**

Need Help From The Big Bhiks!

Nikolas Wolfe <nwolfe@cs.cmu.edu> Wed, Nov 16, 2016 at 3:40 PM
To: Aditya Sharma <adityasharmacmu@gmail.com>
Cc: bhiksha raj <bhiksha@cs.cmu.edu>
Bcc: Don Wolfe <dwolfe314@gmail.com>

Hey comrades,

It's Day 8 in Trump's America. All hail our Great Leader and may rats gently chew on his balls while he sleeps...

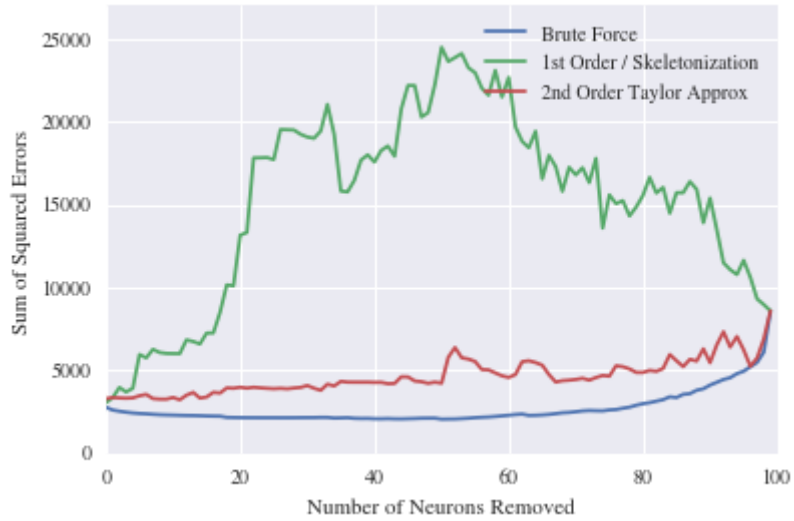
Anyway, looks like the ICLR review process is taking its sweet time. We can keep chipping away... Bhiksha, would you be able to "de-hand-wave" for us a bit? Or at least tell us the limitations of the assertions we're making and how to dial it back if we're overreaching.

http://openreview.net/forum?id=BkV4VS9ll

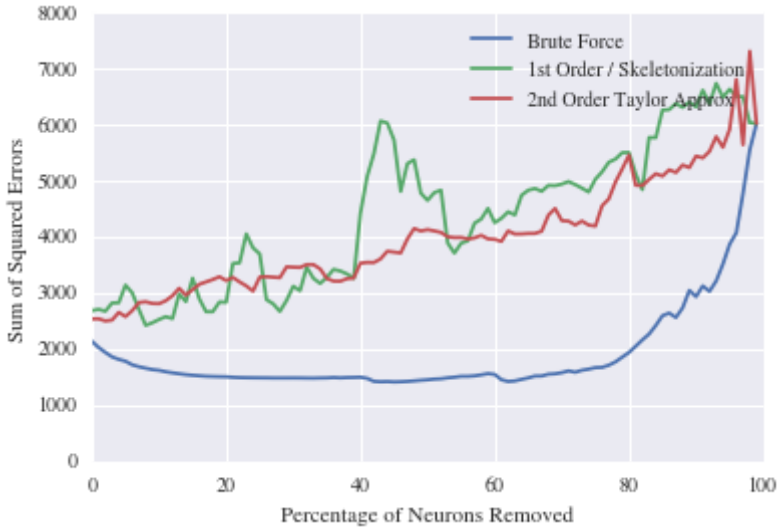
I have some new results / curves for you. This stupid research paper won't die, will it.

I was thinking specifically about these here images... Haven't included this section yet.

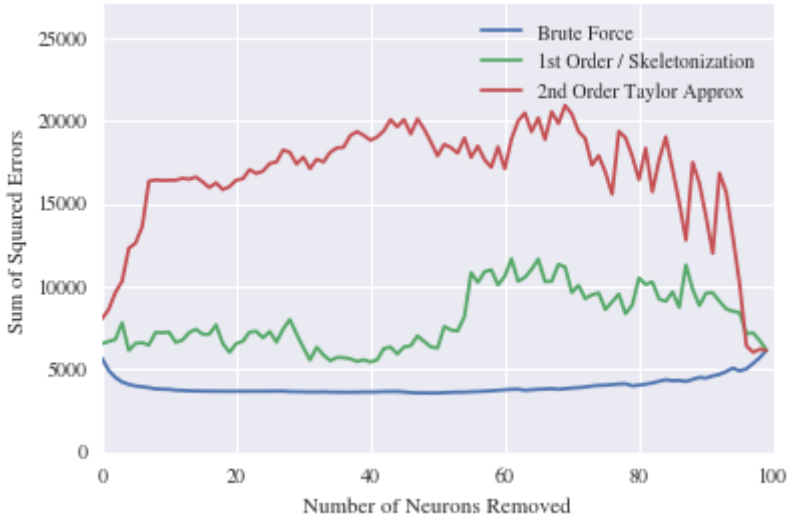
Digit 0:



Digit 1:



Digit 2:



These are the plots when the network didn't learn the function perfectly for digits 0, 1, and 2, respectively (starting accuracies: 0.987, 0.976, 0.93, i think, have to check). input: 784 (full, unnormalized MNIST dataset), output: 2 [yes/no] and hidden layer is dimension 100.

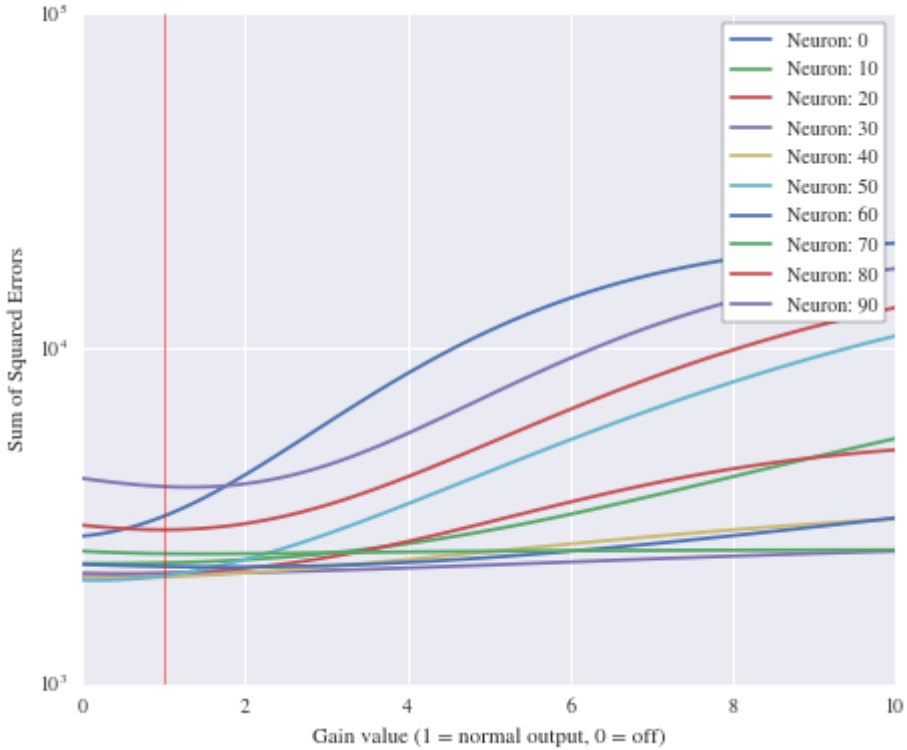
If our hypothesis is correct, what we're seeing here is an effect caused by the imperfect "cancellation" of unimportant nodes. The brute force method picks out the ones engaged in the noise cancellation at first and the output improves, because they weren't helping anyway. This is culling the fat so the nodes that are actually doing the heavy lifting can shine through.

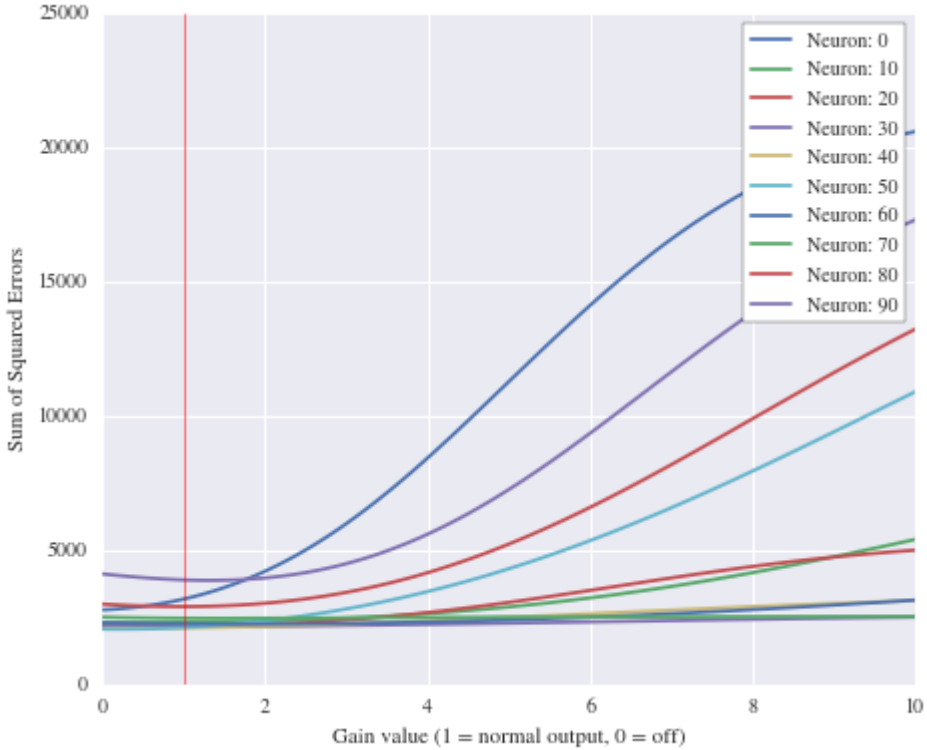
Notice in all three that the blue curve gets better and does not start ticking up until around the 60% mark. This would suggest that the bare minimum required to solve this problem was somewhere in the range of 40-50 neurons. The rest are redundant. It ticks off the "noise cancelling" neurons one by one, and once you force it, it

just has to start getting worse. This is a single-layer network so we expect the error to show a gradual uptick like this, as opposed to being more staggered with a multi-layer network (which is the effect of removing neurons in shallow layer and their impact cascading to deeper neurons).

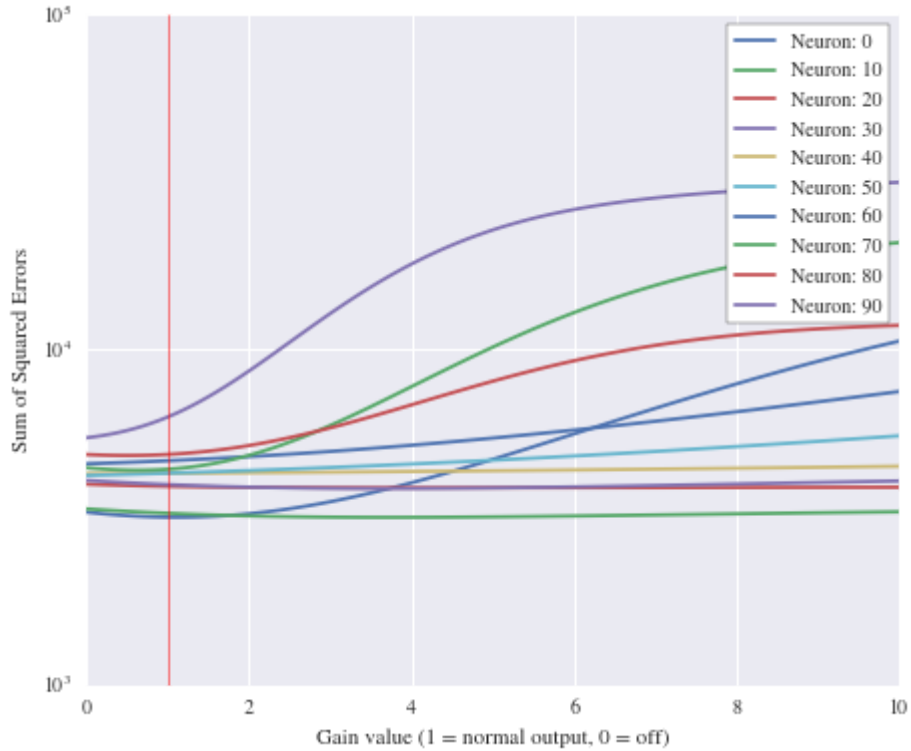
How to explain the red/green? Well, in general I think the first & second order methods they just do a bad job when they don't have a clear signal, i.e. the error surface with respect to a given node is not very flat... Looking at the internal node graphs for 1st/2nd methods from the graphs above we see this:

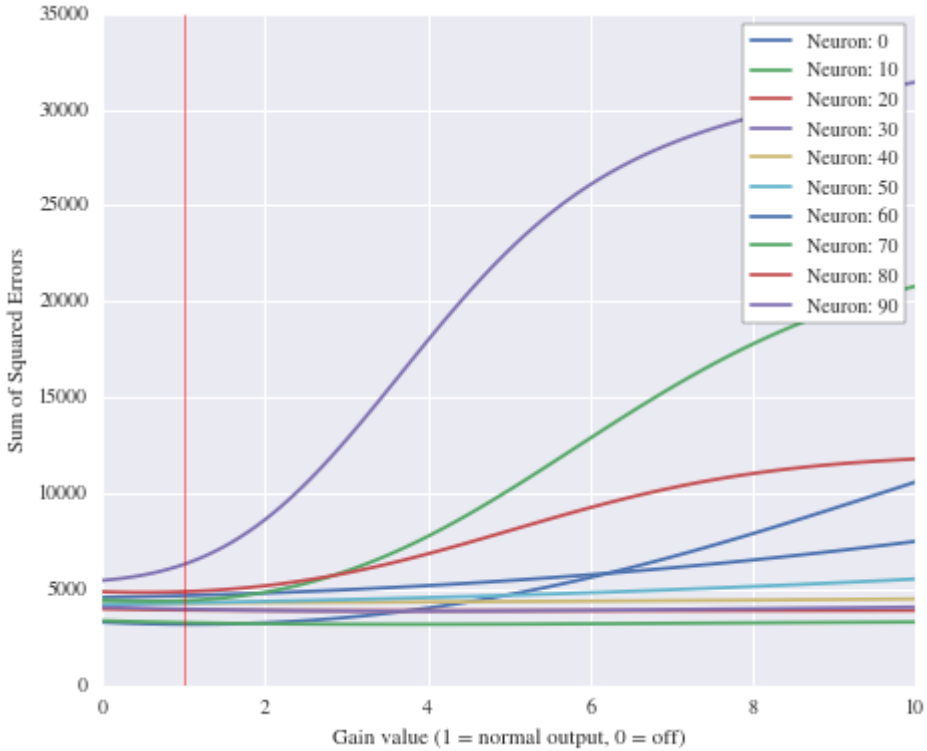
Brute Force: Digit 0:



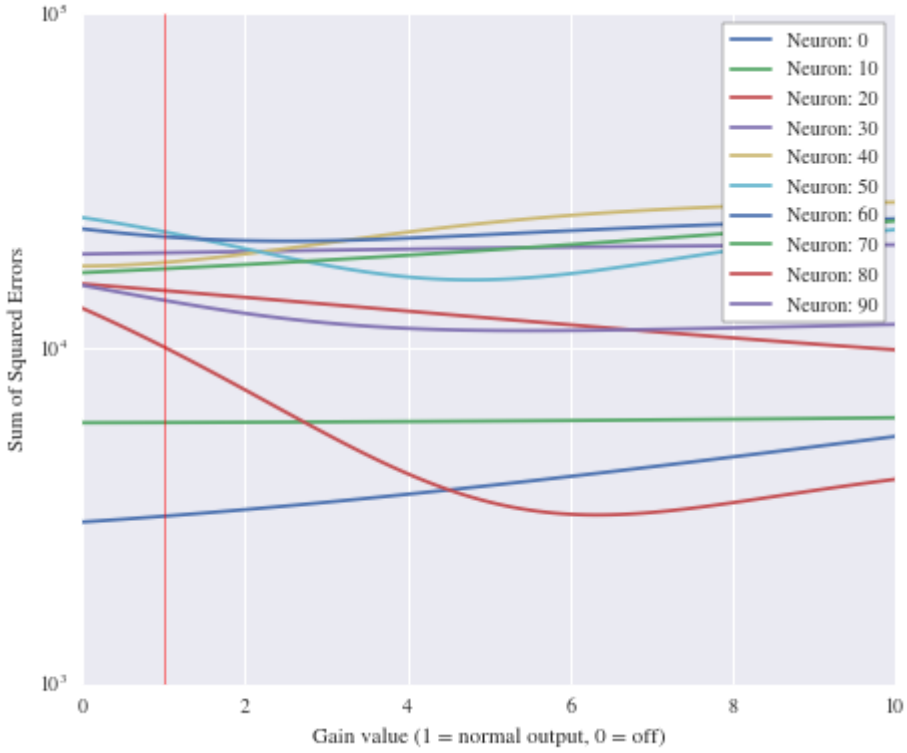


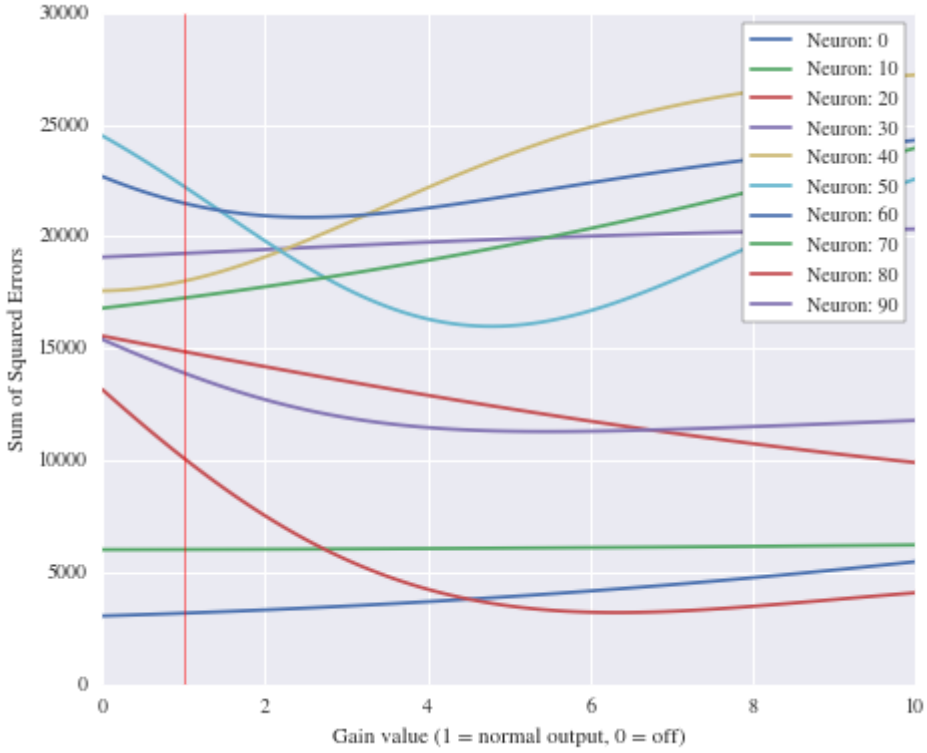
2nd Order Method: Digit 0:



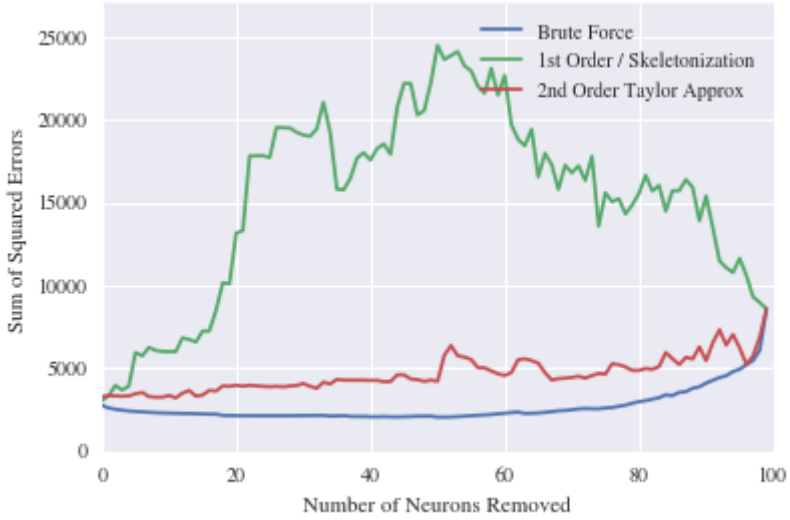


1st Order Method: Digit 0:





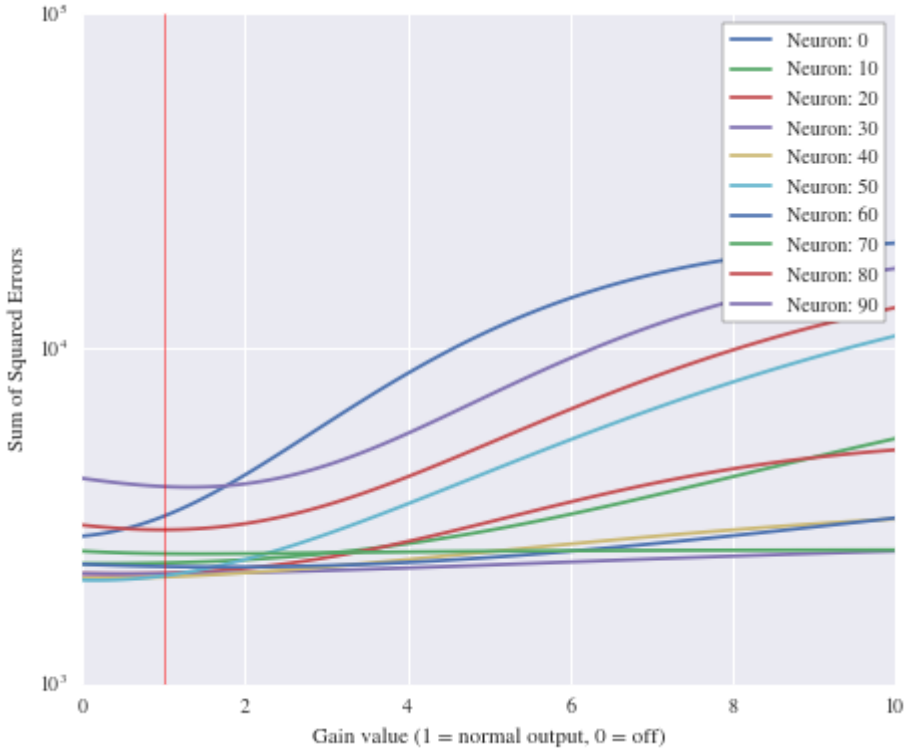
Look at how CLOSE the 2nd order method is to the brute force! Look again at what that gets you:

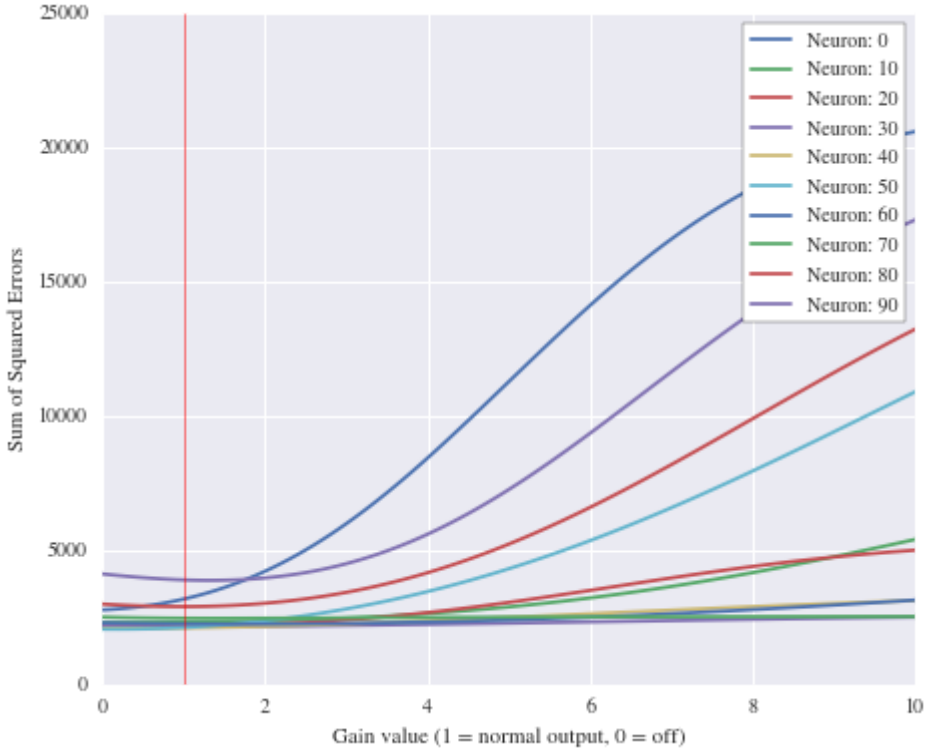


You can see the staggered heights of the curves in the 2nd order method vs. the first. Brute force method is much more collapsed for the first 50 neurons. Now look at how BAD the 1st order method is. **Note now how good a 1st derivative is at approximating the change in error!**

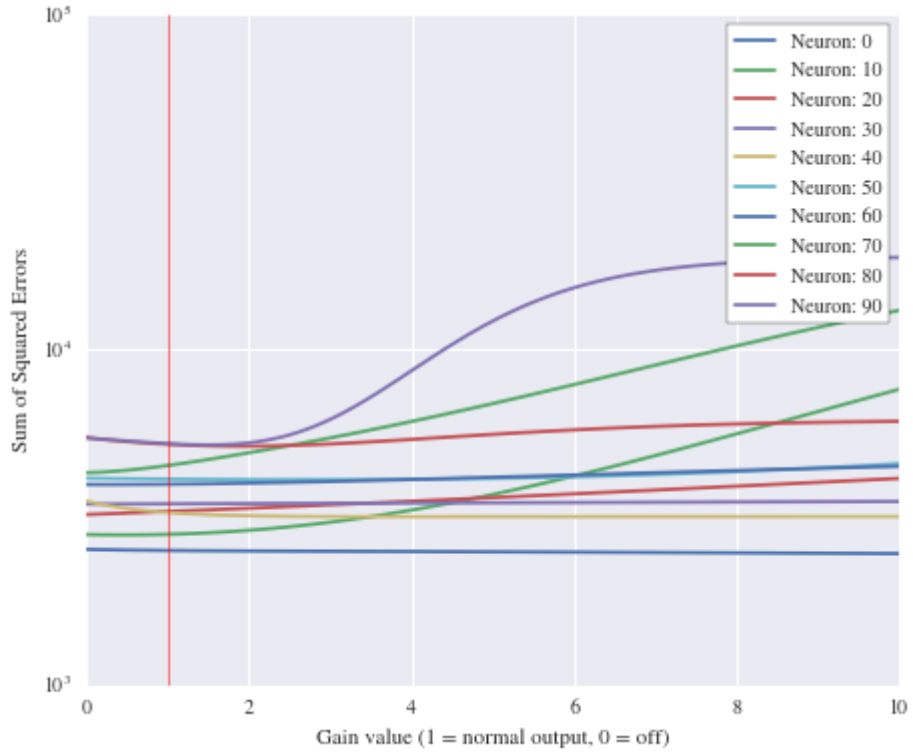
Almost all of these curves are linear from the red line down to 0. Even the log plots show them as being more or less flat. So a Taylor series approximation of the error surface might be accurate, but it **still makes a poor pruning decision.**

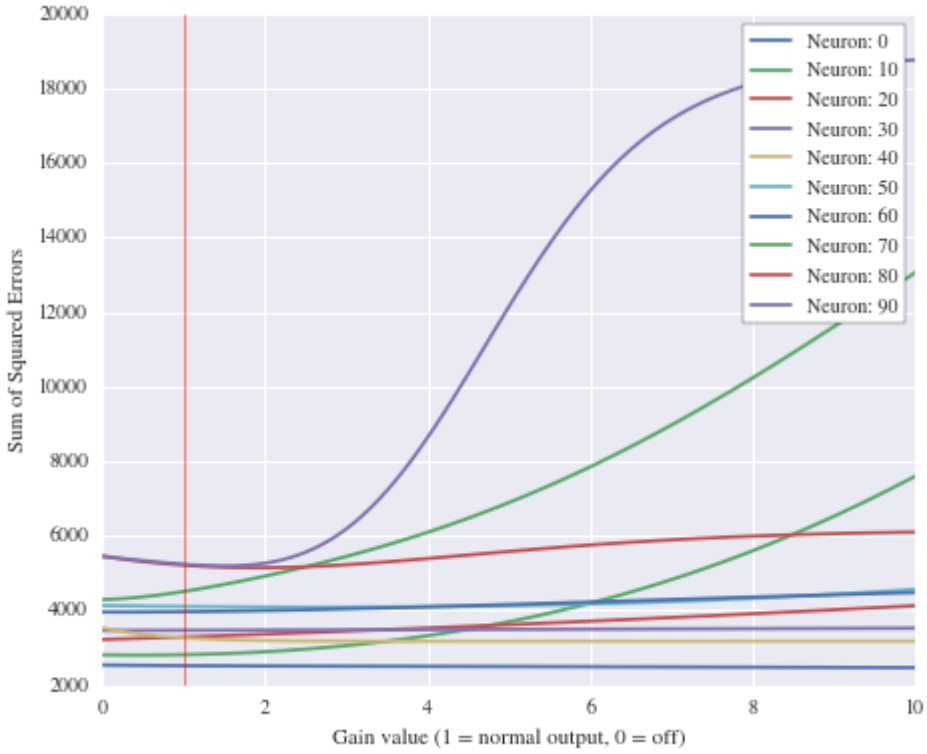
Brute Force: Digit 1:



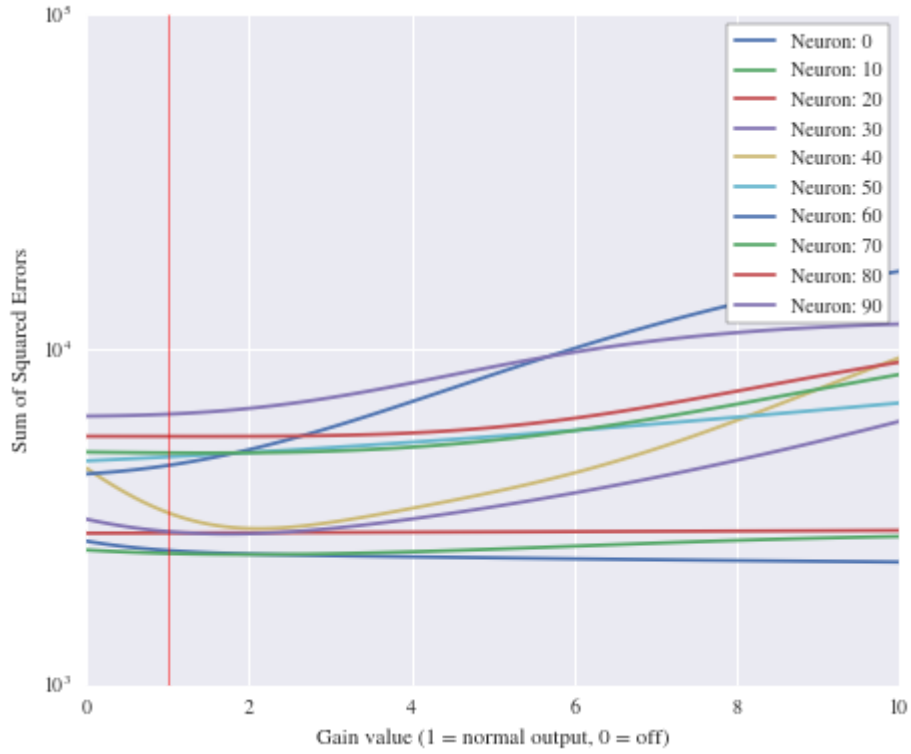


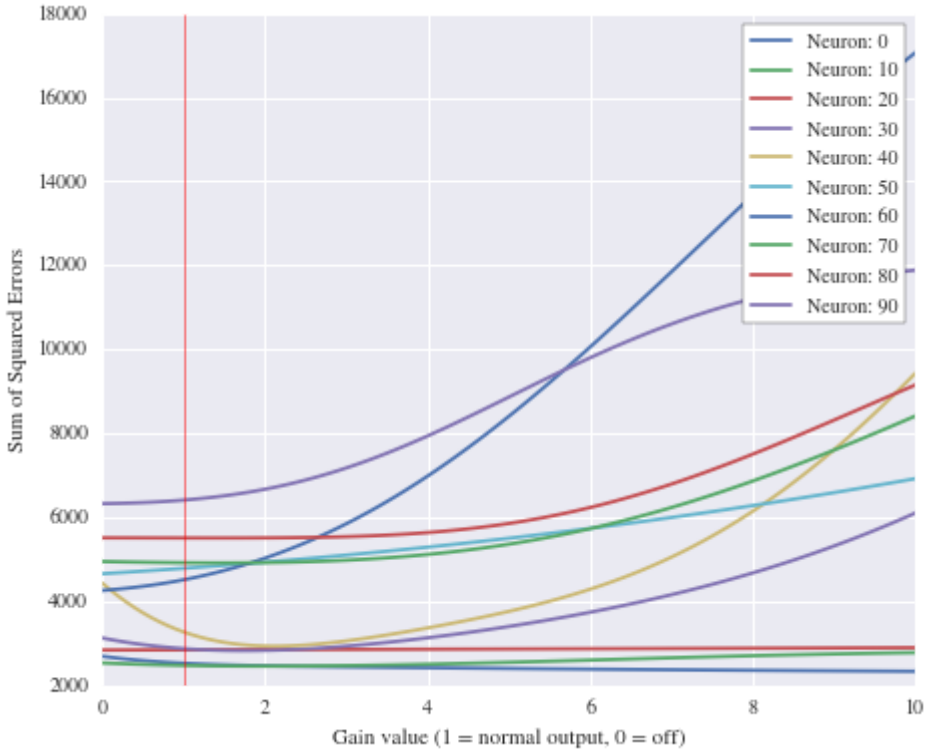
2nd Order: Digit 1:



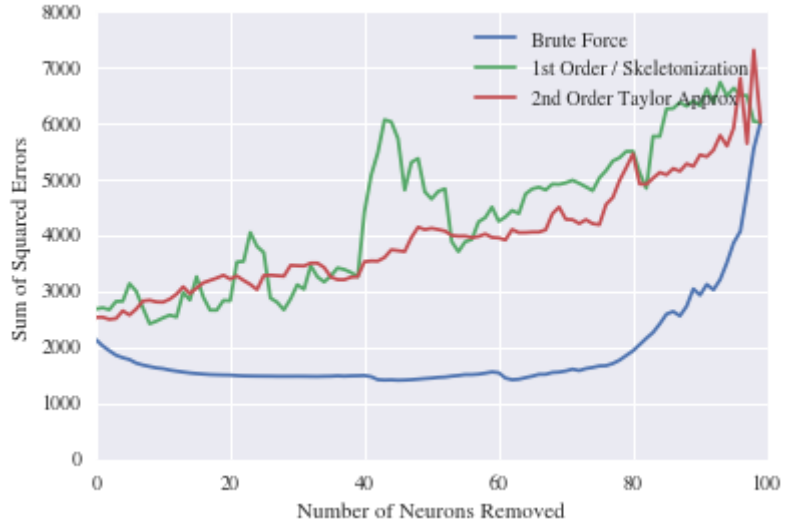


1st Order: Digit 1:



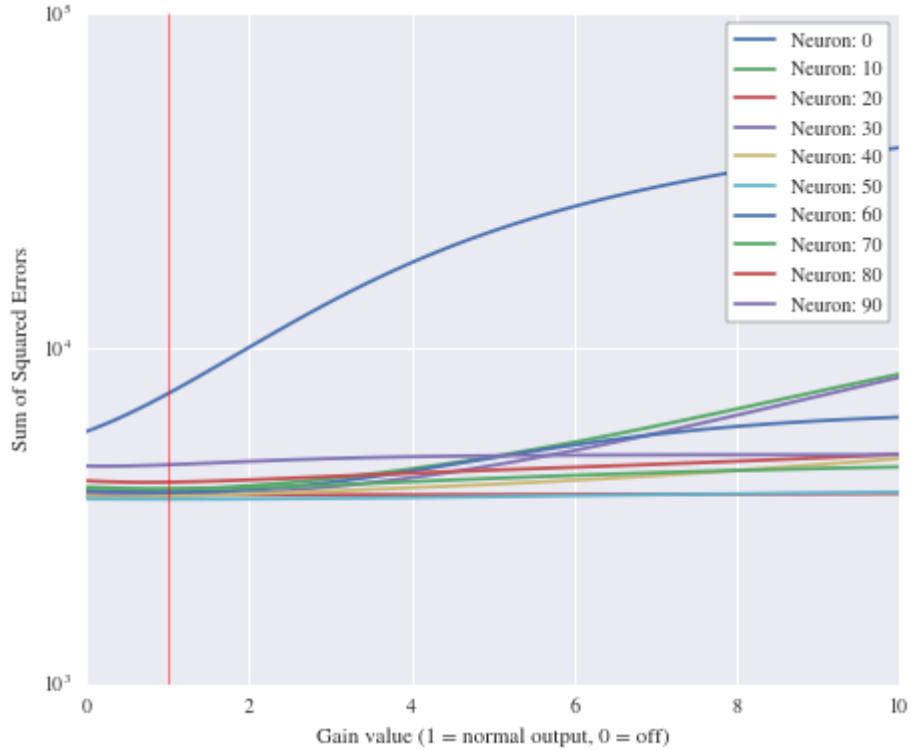


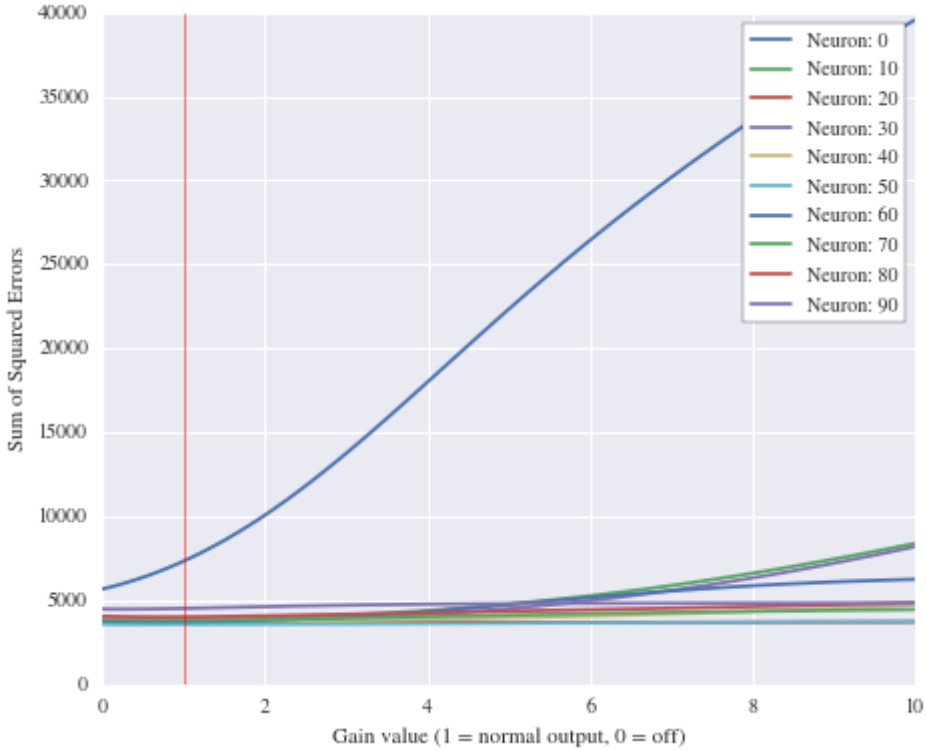
These don't look as bad as before and likewise neither does the plot:



2nd order method is clearly superior... in this case. But the brute force curves are all pretty tight together. Now get ready to have your bubble burst:

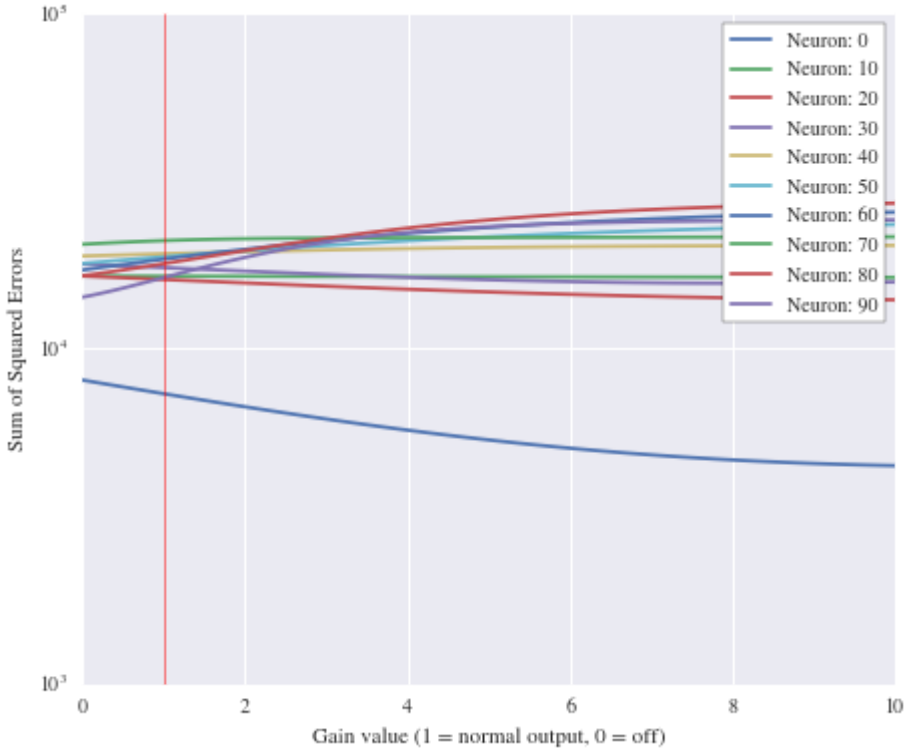
Brute Force: Digit 2:

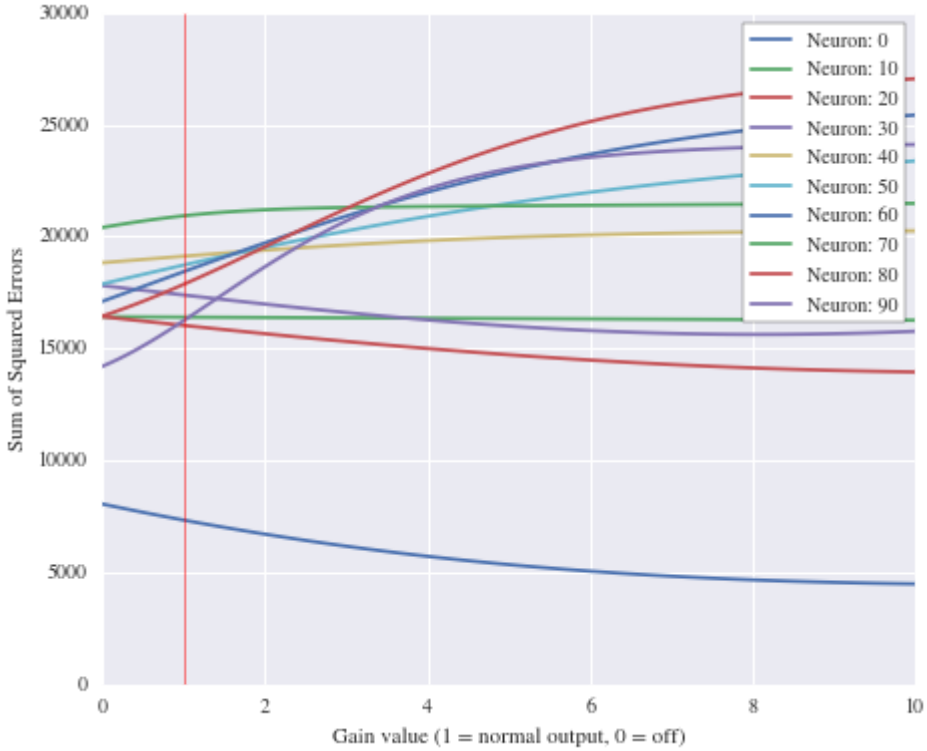




Almost perfect.

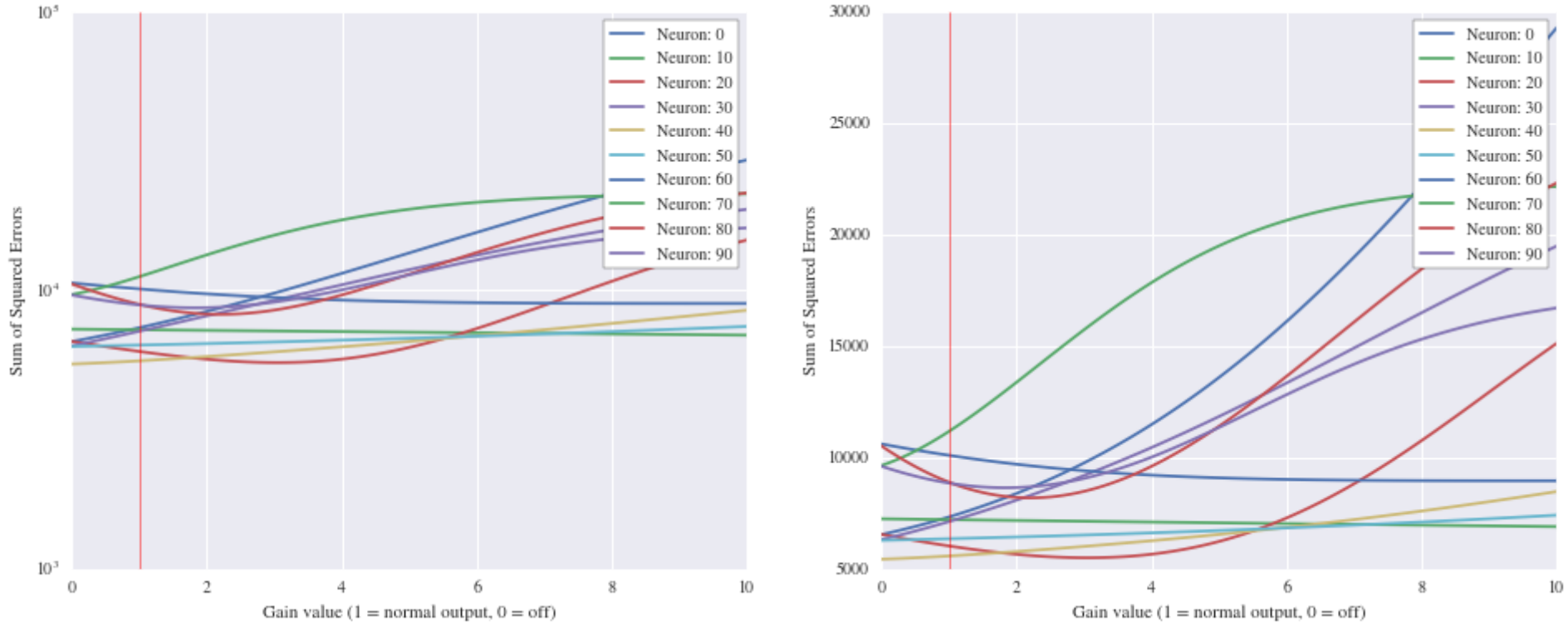
2nd Order: Digit 2:





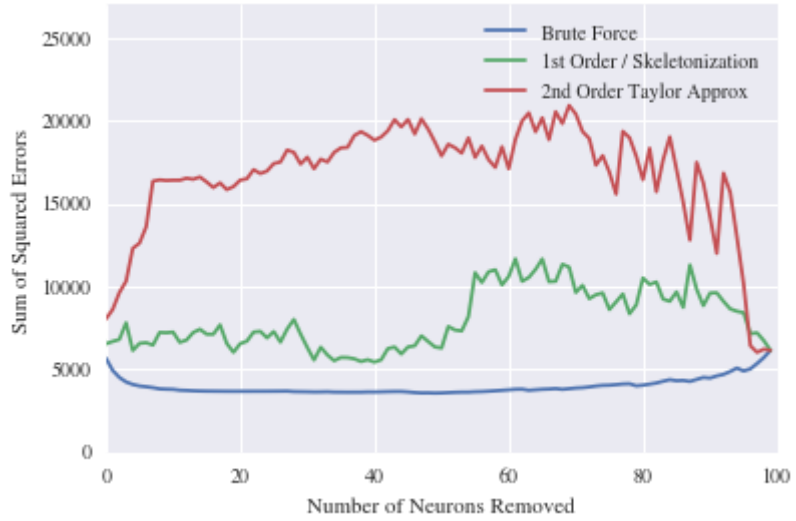
WTF! It appears that 2nd order method seems to have gambled on the hopes that these neurons would "improve" output by being pruned (note that they're all negatively sloped heading towards zero.) Good pruning decisions, from this vantage point. But bad overall.

1st Order: Digit 2:



Better than 2nd order method, in this case. But close only counts with horseshoes and hand grenades.

Here's the plot again:



2nd Order method kicked the bucket in the first try.

It seems that unless the starting network is perfect, there's too much noise going on. Remember there's two parallel function minimizations going on: The input-output function, and the noise introduced by random weight initialization that needs to be smoothed out.

Hoping to spur some discussion with this. I'll write it up in the paper too...

Cheers,

Nik

[Quoted text hidden]