

HW 3

Evan Ott

UT EID: eao466

September 19, 2016

Line search

(A)

The main idea behind the Wolfe conditions are (irrespective of the method of choosing the direction) to ensure that the optimization gets better (sufficient decrease) and stops when it seems like it can't get *much* better (curvature). The backtracking approach (especially thinking about fixed-precision values) takes one huge step, and if the function being optimized isn't better, it moves along the line between the start and the huge step, closer and closer to the start until it is equivalent to where it was or better. In terms of nice properties, it has the capacity to keep making big steps when it's far from the minimum while ensuring that it will always do no worse than it did on the previous iteration. Why? Since the intermediate step-length decreases exponentially, eventually it will be so close to 0 that in the worst case the BLAH BLAH BLAH

Pseudocode:

(B)

Notes from class

Parallel execution in R packages: `foreach`, `parallel`, `doMC`. Good for cross-validation and similar.

Neural synchrony research

Takeaways:

- GLM's are super useful.
- Most problem involve complicated pipelines
 - Massive raw data from a study (like neural recordings)
 - End goal is a model for tiny data
 - Lots of pre-processing, etc. to get to z -scores or something
 - In this project, raw voltage traces \rightarrow ~ 8000 test statistics
 - Each step in that pipeline makes a ton of model choices, which can bias end results

Things like "spike sorting" are crazy complicated, but seen as just a pre-processing step. This is where the electrode picks up more than one neuron but each one seems to be a special snowflake so across many spikes you can actually figure out the mixture, generally speaking.

Take N samples of M electrodes, match up how often cell i fired at time t , get that trial-averaged histogram.

How to tell things like within-trial background changes, stimulus effects, etc.? Point-process models: GLM framework. In this case, ended up being a super-crazy big logistic regression problem. Do that to figure out how likely it *should* be for two particular neurons to fire at the same time. Then check how many *were* seen and this tells you how synchronous they are.

Slap that on a log scale, should get something Normal-ish centered at 0. But it's a little bumpy in some places. Have ~ 8000 test statistics with standard errors, want to correct for false discovery rate. Standard thing would be Benjamini-Hochberg to correct for multiplicity. Problem: we have covariates for each test that we expect to be related to the likelihood of synchrony. BH throws this away. For example, some electrodes are closer together!