Numerical Computation

Alexander G. Ororbia II

Introduction to Machine Learning

CSCI-635

9/8/2023

Overview

- ML algorithms usually require a high amount of numerical computation
  - To update estimate of solutions iteratively
    - not analytically derive formula providing expression
- Common operations:
  - Optimization
    - Determine maximum or minimum of a function
  - Solving system of linear equations
- Just evaluating a mathematical function of real numbers with finite memory can be difficult

Overflow and Underflow

- Problems caused by representing real numbers w/ finite bit patterns
  - For almost all real numbers, we resort to / encounter approximations
- Rounding error(s) might compound across many operations; lead to algorithm failure
  - Numerical errors
    - ***Underflow*** = when numbers close to zero are rounded to zero, e.g., log(0) = $-\infty$ (becomes *NaN*, nont-a-number, in later ops)
    - ***Overflow*** = when numbers w/ large magnitude are approximated as $-\infty$ or ∞ (again, become *NaN*)

Function needing stabilization for Over/Underflow

- Softmax probabilities in multinoulli

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

  - Consider when all x_i are equal to some c. Then all probabilities must equal $1/n$. This may not happen
    - When c is a large negative; denominator $=0$, result undefined underflow
    - When c is large positive, $\exp(c)$ will overflow
  - Circumvented using $\text{softmax}(\boldsymbol{z})$ where $\boldsymbol{z}=\boldsymbol{x}-\max_i x_i$
- Another problem: underflow in numerator can cause $\log \text{softmax}(\boldsymbol{x})$ to be $-\infty$
  - Same trick can be used as for softmax

Dealing with Numerical Considerations

- Developers of low-level libraries should take this into consideration
- ML libraries should be able to provide such stabilization
  - **Libraries such as Tensorflow, Pytorch, Theano detect and safeguard against this (automatically)**

Poor Conditioning

- Conditioning refers to how rapidly a function changes with a small change in input
- Rounding errors can rapidly change the ouput

y = A * x, we want to (pseudo-)invert A, yielding x = A^(-1) * y (e.g., dim reduction)

→ ***Uninvertible*** means we do not have enough data but…

→ A^(-1) is "almost uninvertible" with high condition number

  → Projection/reduction is inaccurate or "garbage"

  → Use eigendecomposition to find condition numbers

QUESTIONS?