Numerical Computation

Sanjay Singh[⋆†]

[⋆]Department of Information and Communication Technology
Manipal Institute of Technology, Manipal University
Karnataka-576104, INDIA
sanjay.singh@manipal.edu

[†]Centre for Artificial and Machine Intelligence (CAMI)
Manipal University, Karnataka-576104, INDIA

February 21, 2017

- ML algorithms requires a high amount of numerical computation
- Numerical methods solves mathematical problems by an iterative process rather than analytical analysis (e.g., Newton-Raphson method)
- Common operations include optimization and solving a system of linear equations
- Evaluating a mathematical function involving real numbers on computer can be difficult using a computer

Overflow and Underflow

- Underflow is a type of rounding error
- It occurs when numbers near zero are rounded to zero
- Many function behave qualitatively differently when their argument is zero rather than a small positive number
- For example, we avoid division by zero (NaN-in MATLAB)

- Overflow is another form of numerical error
- It occurs when numbers with large magnitude are approximated as ∞ or $-\infty$
- Further arithmetic will change these infinite values int NaN (not-a-number) values
- One function that must be stabilized against underflow and overflow is the [1]softmax function
- Softmax function is used to predict the probabilities associated with a multinoulli distribution

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

[1]Softmax function is a generalization of logistic function that “squqshes” a n-dimensional vector $\boldsymbol{x}$ of arbitrary values to a n-dimensional vector softmax$(\boldsymbol{x}) \in [0, 1]$ that adds upto 1.

- What happens when all x_i are some constants c?
- Analytically the output should be $\frac{1}{n}$, but numerically, this may not happen when c has large magnitude
- If c is very large and negative, then $\exp(c)$ will underflow, which means the denominator will become zero, so the final result is undefined (i.e., NaN error)
- If c is very large and positive, $\exp(c)$ will overflow, resulting in the expression as whole being undefined
- Both of these difficulties can be resolved by evaluating softmax(z) where $\boldsymbol{z} = \boldsymbol{x} - max_i x_i$
- The value of softmax function remain unchanged by adding or subtracting a scalar from the input vector
- Subtracting $max_i x_i$ results in the largest argument to $\exp$ being 0, which rules out the possibility of overflow
- Likewise, at least one term in the denominator has a value 1, which rules out the possibility of underflow in the denominator leading to a division by zero

Poor Conditioning

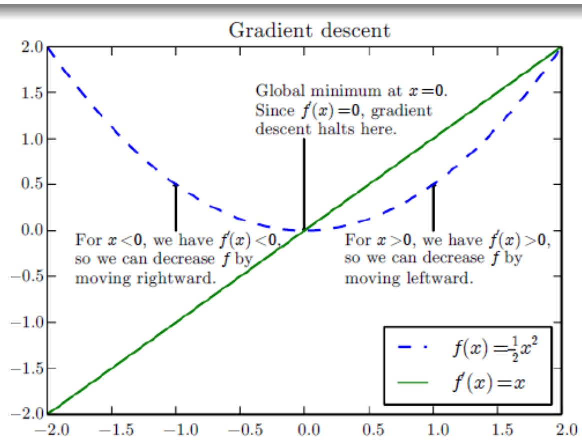
- Conditioning refers to how rapidly a function changes wrt small changes in its inputs
- Ill-conditioned functions are problematic for scientific computation because rounding errors in inputs can result in large changes in the output
- Consider a function $f(x) = A^{-1}x$, when $A \in \mathbb{R}^{n \times n}$ has an eigenvalue decomposition, its condition number is defined as

$$\left| \frac{\lambda_{max}}{\lambda_{min}} \right|$$

- When condition number is large, matrix inversion is sensitive to error in the input
- Such sensitivity is an intrinsic property of the matrix itself, not the result of any rounding error during matrix inversion

Gradient-Based Optimization

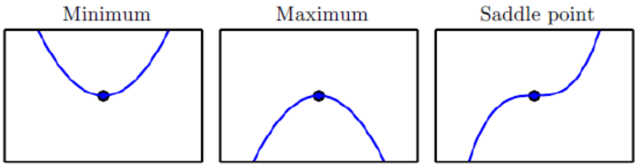
- Most ML or DL algorithms involve optimization of some sort
- Optimization refers to the task of either minimizing or maximizing some function $f(x)$ by altering $\boldsymbol{x}$
- The function we want to minimize or maximize is called objective function or criterion
- When we are minimizing it, we call it the *cost function*, loss function, or error function
- We denote the value that minimize or maximize a function with a superscript *. For example, $x^* = \arg\min f(x)$



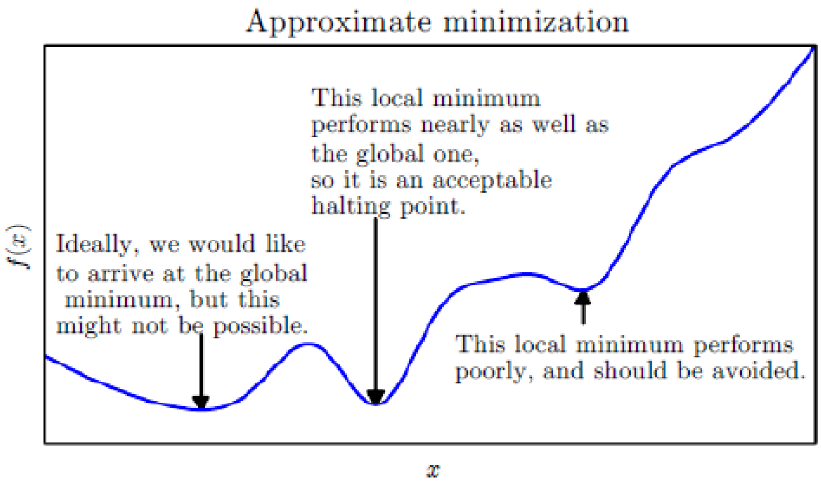
- Consider a function $y = f(x)$, derivative $f'(x)$ gives the slope of $f(x)$ at point x
- In other words, it specifies how to scale a small change in the input in order to obtain the corresponding change in output:

$$f(x+\epsilon) \approx f(x) + \epsilon f'(x)$$

- Derivative is useful for minimizing a function because it tells us how to change x in order to make a small improvement in y



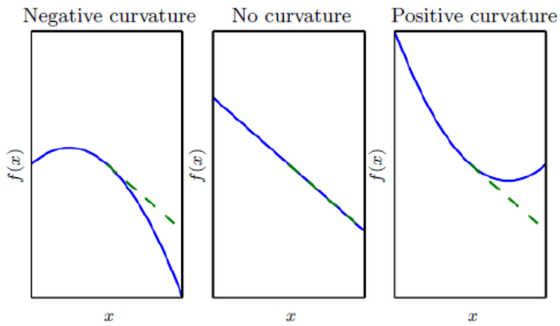
- Points where $f'(x) = 0$ are known as critical points or stationary points
- Critical points that are neither maxima nor minima are known as saddle points



- Gradient points directly uphill, and the negative gradient points downhill
- We can decrease f by moving in the direction of negative gradient
- It is known as the method of steepest descent or gradient descent
- Steepest descent proposes a new point

$$x' = x - \eta \nabla_x f(x)$$

- where η is a learning rate, a positive scalar determining the step size
- There are many ways to choose η
- We can solve for the step size that makes the directional derivative vanish
- Also we can evaluate $f(x - \eta \nabla_x f(x))$ for several values of η and choose one that results in smallest objective function value (also called line search)



- Second derivative determines the curvature of a function
- In case of negative curvature, the cost function decreases faster than the gradient predicts
- In case of no curvature, the gradient predicts the decrease correctly
- In case of negative curvature, the cost function decreases slower than the expected and eventually begins to increase

- When a function has multidimensional input, there are many second derivatives
- We use Hessian matrix, defined as

$$H(f(x))_{i,j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

- Equivalently, Hessian is the Jacobian of the gradient
- Hessian matrix is symmetric (in DL we'll encounter symmetric Hessian everywhere)
- Hessian matrix is decomposable into eigenvectors and eigenvalues

- Second derivative in a specific direction, represented by vector $\boldsymbol{d}$ is given by $d^T Hd$
- When d is an eigenvector of H, second derivative in that direction is the corresponding eigenvalue
- The second derivative tells us how well we can expect a gradient descent step to perform
- We can make a second-order Taylor series approximation to the function $f(x)$ around the current point $x^{(0)}$:

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T g + \frac{1}{2}(x - x^{(0)})^T H(x - x^{(0)})$$

  where g is the gradient and H is the Hessian at $x^{(0)}$
- New point x is given by $x^{(0)} - \eta g$, substituting this in our approximation, we get

$$f(x^{(0)} - \eta g) \approx f(x^{(0)}) - \eta g^T g + \frac{1}{2}\eta^2 g^T Hg$$

$$f(x^{(0)} - \eta g) \approx f(x^{(0)}) - \eta g^T g + \frac{1}{2}\eta^2 g^T Hg$$

- There are three terms:
  1. original value of the function
  2. the expected improvement due to the slope function, and
  3. the correction we must apply to account for the curvature of the function
- When the last term is large GD move uphill
- When $g^T Hg$ is zero or negative, the Taylor series approximation predicts that increasing η will decrease f
- When $g^T Hg$ is positive
- Step size η that will decrease Taylor series approximation of the function is given by

$$\eta^* = \frac{g^T g}{g^T Hg}$$

- When g aligns with the eigenvector of H corresponding to the maximal eigen value λ_{max}, then $\eta^* = \dfrac{1}{\lambda_{max}}$

- Second derivative test
  - When $f'(x) = 0$, and $f''(x) > 0$, we conclude that x is a local minimum
  - When $f'(x) = 0$, and $f''(x) < 0$, we conclude that x is a local maximum
  - Test is inconclusive for $f''(x) = 0$, we may consider x as a saddle point or a part of flat region
- We can generalize second derivative test in higher dimension as
  - At a critical point, $\nabla_x f(x) = 0$, one can examine eigen values of Hessian matrix
  - When the Hessian is positive definite (i.e., all eigenvalues are positive), the point is a local minimum
  - When the Hessian is negative definite (i.e., all eigenvalues are negative), the point is a local maximum

Newton's Method

- It is based on second-order Taylor series expansion to approximate $f(x)$ aroud $x^{(0)}$

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T \nabla_x f(x^{(0)}) + \frac{1}{2}(x - x^{(0)})^T H(f)(x^{(0)})(x - x^{(0)})$$

- On solving for critical point, we obtain

$$x^* = x^{(0)} - H^{-1}(f)(x^{(0)}) \nabla_x f(x^{(0)})$$

- When f is PD quadratic function, applying Newton's methods once yields the minimum of the function directly
- When f is not truly quadratic but can be locally approximated as PD, one need to apply Newton's method multiple times

- Optimization algorithms such as gradient descent that use only the gradient are called first-order optimization
- Optimization algorithms such as Newton's method that also use Hessian are called second-order optimization algorithms
- Optimization algorithms employed in DL are applicable to a wide variety of functions, but comes with no guarantees
- We gain some guarantee by restricting to functions that are either Lipschitz continuous or have Lipschitz continuous derivative
- A Lipschitz continuous function is a function f whose rate of change is bounded by a Lipschitz constant $\mathcal{L}$:

$$\forall x, y, \|f(x) - f(y)\| \leq \mathcal{L} \, \|x - y\|$$

- Such assumption allows us to quantify that a small change in the input made by an algorithm will have small change in the output
- Lipschitz continuity ensures that function f is not ill-conditioned
- Lipschitz constraints is a fairly weak constraints
- Convex optimization algorithms are able to provide more guarantees by making stronger restrictions
- Convex optimization algorithms are applicable only to convex function-functions for which Hessian is PSD everywhere
- Convex function don't have saddle points and all of their local minima are global minima