Department of AI, University of Seoul
Machine Learning and Artificial Intelligence Lab

Deep Learning

Numerical Computation

Kyungwoo Song

Numerical Computation

Overflow and Underflow

Overflow and Underflow

- We need to represent infinitely many real numbers with a finite number of bit patterns
- For almost all real numbers, we incur some approximation error

```
import numpy as np
import torch
```

```
[48] z = 65504 * 2
x1 = np.asarray(1.0+z,dtype=np.float16)
x2 = np.asarray(1.0+z,dtype=np.float32)
x3 = np.asarray(1.0+z,dtype=np.float64)
print(x1,x2,x3)
```

```
inf 131009.0 131009.0
```

- Underflow
  - Numbers near zero are rounded to zero
  - $\frac{1}{x}, \log x, \ldots$
- Overflow
  - Overflow occurs when number with large magnitude are approximated as ∞ or $-\infty$

```
[50] z = 3.4e39
x1 = np.asarray(1.0+z,dtype=np.float16)
x2 = np.asarray(1.0+z,dtype=np.float32)
x3 = np.asarray(1.0+z,dtype=np.float64)
print(x1,x2,x3)
```

```
inf inf 3.4e+39
```

```
[53] z = 1.8e308
x1 = np.asarray(1.0+z,dtype=np.float16)
x2 = np.asarray(1.0+z,dtype=np.float32)
x3 = np.asarray(1.0+z,dtype=np.float64)
print(x1,x2,x3)
```

```
inf inf inf
```

Source:

Softmax function

Overflow and Underflow

- Softmax function
  - $softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$
- When all of the x_i are equal to some constant c
  - If c is very negative, $\exp(c)$ will underflow
  - If c is very large and positive, $\exp(c)$ will overflow
- Simple algebra solves the numeric problem
  - $softmax(x)_i = softmax(x + c)_i$
  - $z = x - \max_i x_i$
    - ❖Largest argument to exp being 0
    - ❖=> rules out the possibility of overflow in the numerator
    - ❖=> rules out the possibility of underflow in the denominator

Source:

Poor Conditioning

- $\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$
  - $= \max_{\|x\|=1} \|Ax\|_p$ $(\|cx\| = \|c\|\|x\|)$
  - maximum amount by which Ax can lengthen any unit-norm input
- For $p = 2$, $\|A\|_p = \sqrt{\lambda_{\max(A^T A)}} = \max_i \sigma_i$ (The largest singular value)
  - σ_i: singular value
- Why?
  - $\sup_{\|x\|_2=1} \|Ax\|_2 = \sup_{\|x\|_2=1} \|U\Sigma V^T x\|_2$ (by SVD)

$$= \sup_{\|x\|_2=1} \|\Sigma V^T x\|_2$$

$$= \sup_{\|y\|_2=1} \|\Sigma y\|_2 \ (y = V^T x)$$

$\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_n)$
The maximum is attained when $y = (1, \dots, 0)^T$

❖ U and V are unitary matrix, $U^T = U^{-1}$ (정확하게는 conjugate transpose)

❖ $\|Ux_0\|_2^2 = x_0^T U^T U x_0 = x_0^T x_0 = \|x_0\|_2^2$ for some vector x_0

❖ $\|y\|_2 = \|V^T x\|_2 = \|x\|_2 = 1$

Source: https://math.stackexchange.com/questions/586663/why-does-the-spectral-norm-equal-the-largest-singular-value

Condition Number

Poor Conditioning

- Condition number of a matrix A
  - How numerically stable any computations involving A will be
  - $\kappa(A) = ||A|| \cdot ||A^{-1}||$
    - $\kappa(A) \geq 1$
    - $||A||$: l_2 norm unless stated otherwise
- A is well-conditioned if $\kappa(A)$ is small
- A is ill-conditioned if $\kappa(A)$ is large
  - A large condition number means A is nearly singular (non-invertible)
    - A better measure of nearness to singularity than the size of the determinant
  - Example) $A = 0.1I_{100\times 100}$
    - det(A) = 10^{-100}
    - $\kappa(A) = 1$
      - A is well-conditioned (Ax is stable)

$\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}}$ for l_2 norm

- $\sigma_{max}(A^{-1}) = 1/\sigma_{min}(A)$
- The ratio of the largest to smallest singular values

Source: Murphy Book Chapter 7

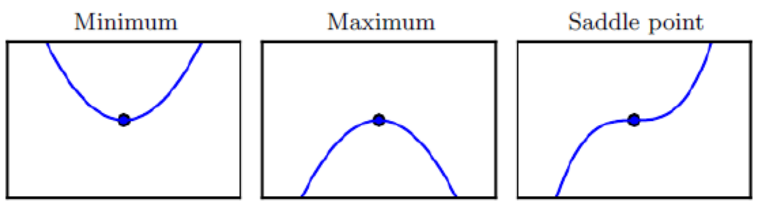
Poor Conditioning

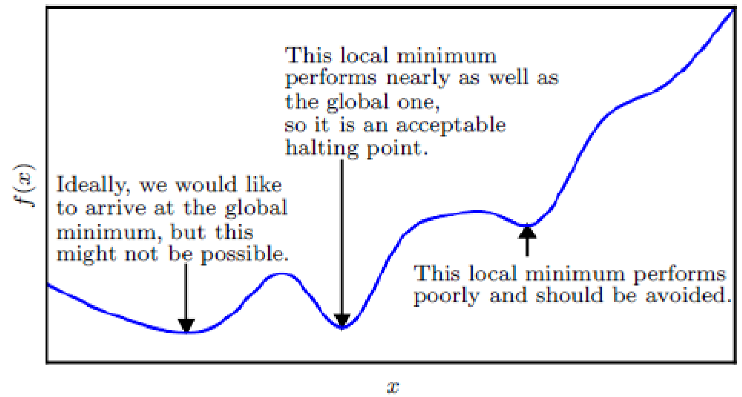
- Consider a linear system of equations
  - $Ax = b$
  - If A is non-singular, the unique solution is $x = A^{-1}b$
  - Suppose we change b to $b + \Delta b$
  - The new solution must satisfy
    - ❖ $A(x + \Delta x) = b + \Delta b$
      - ➢ $\Delta x = A^{-1}\Delta b$
- A is well-conditioned if a small Δb results in a small Δx
- Example) $A = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 + 10^{-10} & 1 - 10^{-10} \end{pmatrix}$, $A^{-1} = \begin{pmatrix} 1 - 10^{10} & 10^{10} \\ 1 + 10^{10} & -10^{10} \end{pmatrix}$
  - Solution for $b = (1,1)$ is $x = (1,1)$
  - $\Delta x = A^{-1}\Delta b = \begin{pmatrix} \Delta b_1 - 10^{10}(\Delta b_1 - \Delta b_2) \\ \Delta b_1 + 10^{10}(\Delta b_1 - \Delta b_2) \end{pmatrix}$
    - ❖ Small change in b lead to an extremely large change in x
  - $\kappa(A) = 2 \times 10^{10}$ => ill-conditioned

Source:

- Optimization
  - Minimizing or maximizing some function $f(x)$ by altering x
  - The function we want to minimize or maximize: objective function or criterion
  - When we are minimizing it, we also call it
    - Cost function, loss function, error function
  - $x^* = argmin\ f(x)$
- Suppose we have a function $y = f(x)$
  - Derivative of this function: $f'(x)$, $\frac{dy}{dx}$
  - How to scale a small change in the input in order to obtain the corresponding change in the output: $f(x+\epsilon) \approx f(x) + \epsilon f'(x)$
  - $f(x - \epsilon\ sign(f'(x)))$ is less than $f(x)$ for small enough ϵ
    - We can thus reduce $f(x)$ by moving x in small steps with opposite sign of the derivative
    - Gradient descent

Source:





- $f'(x) = 0$
  - Points where $f'(x) = 0$ are known as critical points or stationary points
  - Local minimum: points where $f(x)$ is lower than at all neighboring points
  - Local maximum: points where $f(x)$ is higher than at all neighboring points
  - Saddle points: neither maxima nor minima
  - Global minimum: A points that obtains the absolute lowest value of $f(x)$
- In context of deep learning, there are many local minima and many saddle points

Source: Deep Learning Book

Directional Derivative

Gradient-Based Optimization

- For functions with multiple inputs, $\frac{\partial}{\partial x_i} f(x)$
  - How f changes as only the variable x_i increases at point x
  - Gradient: derivative with respect to a vector, $\nabla_x f(x)$
- Directional Derivative
  - Directional derivative in direction u (a unit vector) is the slope of the function f in direction u
  - Derivative of the function $f(x + \alpha u)$ w.r.t. α evaluated at $\alpha = 0$
  - $\frac{a}{\partial \alpha} f(x + \alpha u) = u^T \nabla_x f(x)$ when $\alpha = 0$
  - We would like to find the direction in which f decreases the fastest
    - $\min_{u, u^T u = 1} u^T \nabla_x f(x) = \min_{u, u^T u = 1} \|u\|_2 \|\nabla_x f(x)\|_2 \cos \theta$
      - θ: angle between u and the gradient
      - It simplifies to $\min_u \cos \theta$
      - It is minimized when u points in the opposite direction as the gradient.

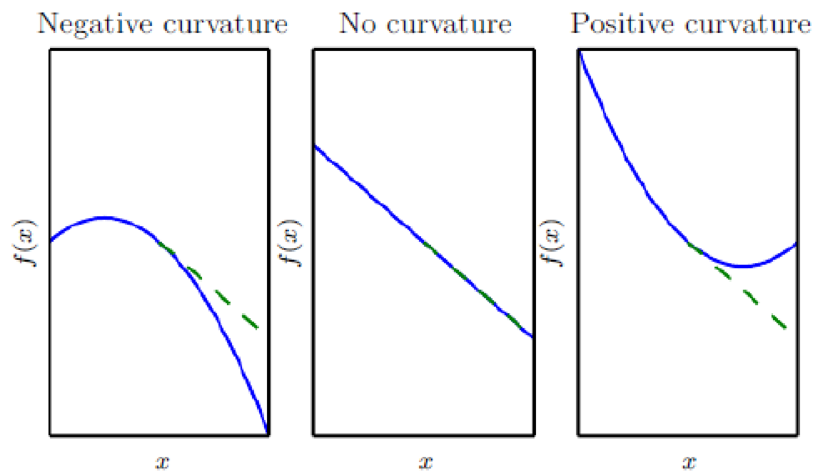
Source:

Steepest descent or Gradient descent

Gradient-Based Optimization

- Directional Derivative
  - $\frac{a}{\partial \alpha} f(x + \alpha u) = u^T \nabla_x f(x)$ when $\alpha = 0$
  - We would like to find the direction in which f decreases the fastest
    - $\min_{u, u^T u = 1} u^T \nabla_x f(x) = \min_{u, u^T u = 1} \|u\|_2 \|\nabla_x f(x)\|_2 \cos\theta$
      - θ: angle between u and the gradient
      - It simplifies to $\min_u \cos\theta$
      - It is minimized when u points in the opposite direction as the gradient.
- $x' = x - \epsilon \nabla_x f(x)$
  - ϵ: learning rate, positive scalar
    - 1) set ϵ to a small constant
    - 2) Solve the 1d minimization problem, $\epsilon_t = argmin_{\epsilon>0} L(x - \epsilon \nabla_x f(x))$
    - 3) evaluate $f\left(x - \epsilon \nabla_x f(x)\right)$ for several values of ϵ, and choose the one

Source:



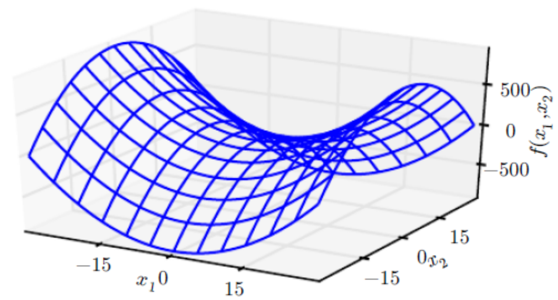
- Gradient
  - $f: R^n \to R$
  - $\nabla f(x)$: $1 \times n$ matrix
  - $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right)$
- Jacobian matrix
  - $f: R^m \to R^n$
  - Jacobian matrix, $J \in R^{n \times m}$
  - $J_{ij} = \frac{\partial}{\partial x_j} f(x)_i$
- Hessian Matrix
  - Second derivative (measuring curvature)
  - Jacobian of the gradient
  - $f: R^n \to R$
  - $H(f)(x)_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(x) = \frac{\partial^2}{\partial x_j \partial x_i} f(x) \Rightarrow$ Hessian matrix is real and symmetric

Source:

- Second derivative
  - Tells us how well we can expect a gradient descent step to perform
  - $f(x) \approx f\left(x^{(0)}\right) + \left(x - x^{(0)}\right)^T g + \frac{1}{2}\left(x - x^{(0)}\right)^T H(x - x^{(0)})$
    - ❖Taylor series approximation
    - ❖g: gradient, H: Hessian at $x^{(0)}$
  - If we use a learning rate of ϵ, the new point will be given by $x^{(0)} - \epsilon g$
  - $\Rightarrow f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^T g + \frac{1}{2}\epsilon^2 g^T H g$
    - ❖$f\left(x^{(0)}\right)$: original value of the function
    - ❖$\epsilon g^T g$: expected improvement due to the slope of the function
    - ❖$\frac{1}{2}\epsilon^2 g^T H g$: correction we must apply to account for the curvature of the function
    - ❖If $\frac{1}{2}\epsilon^2 g^T H g$
      - ➢ is too large, the gradient descent step move uphill
      - ➢ is zero or negative, increasing ϵ will decrease f forever (under the Taylor approximation)
    - ❖Optimal step size: $\epsilon^* = \frac{g^T g}{g^T H g}$ (if $g^T H g$ is positive)

Source:

- Second derivative
  - Determine whether a critical point is a local maximum, minimum, saddle point
  - Critical Point, $f'(x) = 0$
    - ❖When $f''(x) > 0$: local minimum
    - ❖When $f''(x) < 0$: local maximum
    - ❖When $f''(x) = 0$: saddle point or a flat region



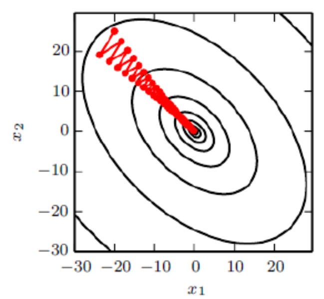
  - Multiple dimension 에서도 동일하게 가능합니다.
    - ❖Second derivative in a specific direction:
    - ❖Hessian matrix is real and symmetric => Eigendecomposition
    - ❖When the Hessian is positive definite: local minimum
      - ➢ All eigenvalues are positive
    - ❖When the Hessian is negative definite: local maximum
      - ➢ All eigenvalues are negative
    - ❖When the Hessian has at least one positive eigenvalue and negative eigenvalue: Saddle Point

Source:

- Second derivative
  - Positive eigenvalue
    - If x is an eigenvector of A then $x \neq 0$ and $Ax = \lambda x$.
    - $x^T A x = \lambda x^T x$
    - If $\lambda > 0$, then as $x^T x > 0$ we must have $x^T A x > 0$
  - Positive definite
    - $x^T A x > 0$ for all vectors $x \neq 0$
  - 특정한 방향에 대한 second derivative
    - $d^T H d$ where d is unit vector
    - 얼마나 서로 second derivative가 다른지 측정 ⇒ Condition number of the Hessian
    - When the Hessian has a poor condition number, gradient descent performs poorly
      - The derivative increases rapidly in one direction, while in another direction, it increases slowly
      - Gradient descent is unaware of this change in the derivative

아니 교수님.
갑자기 왜 전체 vector로 확장 된 것이죠?
Eigenvectors of a symmetric $n \times n$ matrix span all of R^n
⇒ any vector can be represented as a linear combination of the eigenvectors

Condition number: 5
The most curvature: $[1,1]^T$
The least curvature: $[1,-1]^T$



Source: https://www.math.utah.edu/~zwick/Classes/Fall2012_2270/Lectures/Lecture33_with_Examples.pdf