Numerical Computation

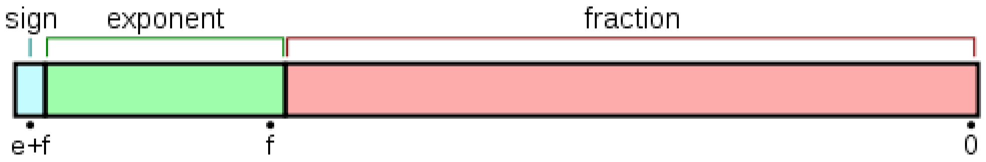
Yen-Chi Chen

2018-10-18

Overflow & Underflow

The root of all evil

- IEEE 754

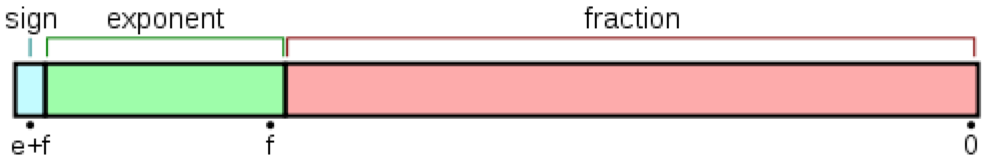


  - $\text{value} = \text{sign} \times 1.\text{fraction} \times 2^{\text{exponent}}$
    - Except for 0
  - For example:
    $-12345 = -1 \times 1.2345 \times 10^4$

  - Memory is finite!!!

The root of all evil

- IEEE 754



  - value $= \text{sign} \times 1.\text{fraction} \times 2^{\text{exponent}}$
    - Except for 0
  - For example:

    $-12345 = -1 \times 1.2345 \times 10^{4}$

  - Memory is finite!!!

Underflow

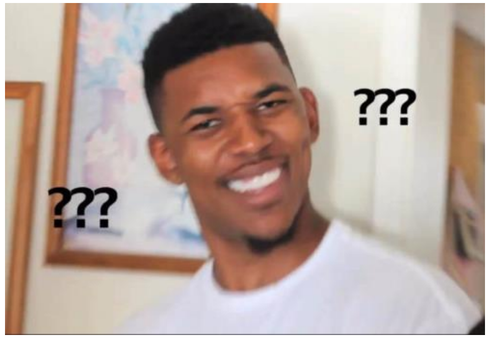
- Too close to zero

Underflow

- Too close to zero
- Min of normal number
  - $\pm 2^{-126} \approx \pm 3.4 \times 10^{38}$

Underflow

- Too close to zero
- Min of normal number (hardly)
  - $\pm 2^{-126} \approx \pm 3.4 \times 10^{38}$



Underflow

- Too close to zero
- Min of normal number (hardly)
  - $\pm 2^{-126} \approx \pm 3.4 \times 10^{38}$
- Computation error   (usually)
  - $a + b - a$,where $a >>>> b$

$1000 + 0.05 - 1000$
$= 1000.05 - 1000$
$= 1.00005 \times 10^3 - 1000$
$= 1.0000 \times 10^3 - 1000$
$= 0$

Underflow

- Too close to zero
- Min of normal number (hardly)
  - $\pm 2^{-126} \approx \pm 3.4 \times 10^{38}$
- Computation error (usually)
  - $a + b - a$,where $a >>>> b$

$1000 + 0.05 - 1000$
$= 1000.05 - 1000$
$= 1.00005 \times 10^3 - 1000$
$= 1.0000 \times 10^3 - 1000$
$= 0$

- $a - a + b$

$1000 - 1000 + 0.05$
$= 0 + 0.05$
$= 0.05$

Overflow

- numbers with large magnitude
  - $-\infty$ & $+\infty$
- Undefined: NaN (Not-a-number)
  - 0/0
  - $(\pm\infty)/(\pm\infty)$
  - $\infty - \infty$
  - $0 * \infty$
  - $\sqrt{-1}$ (non-real number
  - …

Softmax

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

- If $\forall i, \quad x_i = c$, then $softmax(\boldsymbol{x})_i = \frac{1}{n}$

Softmax

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

- If $\forall i, \quad x_i = c$, then $softmax(\boldsymbol{x})_i = \frac{1}{n}$
- When c is very negative
  - $\exp(c) = 0$ ←underflow
  - $softmax(\boldsymbol{x})_i = 0/0 = NaN$ ←undefined

Softmax

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

- If $\forall i,\ \ x_i = c$, then $softmax(\boldsymbol{x})_i = \frac{1}{n}$
- When c is very negative
  - $\exp(c) = 0$ ←underflow
  - $softmax(\boldsymbol{x})_i = 0/0 = NaN$ ←undefined
- When c is very large and positive
  - $\exp(c) = \infty$ ←overflow
  - $softmax(\boldsymbol{x})_i = \infty/\infty = NaN$ ←undefined

Solution

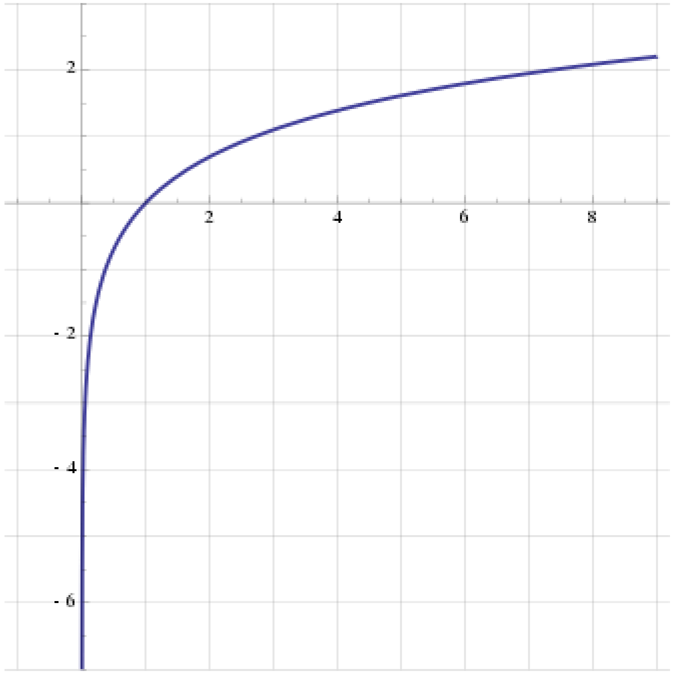
$$softmax(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

- Let $\boldsymbol{z} = \boldsymbol{x} - \max_i x_i$
- Then $softmax(\boldsymbol{z})_i = softmax(\boldsymbol{x})_i$
  - Proof:
  - Let $m = \max_i x_i$

$$softmax(\boldsymbol{z})_i = \frac{\exp(x_i - m)}{\sum_{j=1}^{n} \exp(x_j - m)} = \frac{\exp(x_i) \,/\, \exp(m)}{\sum_{j=1}^{n} \exp(x_j) \,/\, \exp(m)} = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = softmax(\boldsymbol{x})_i$$

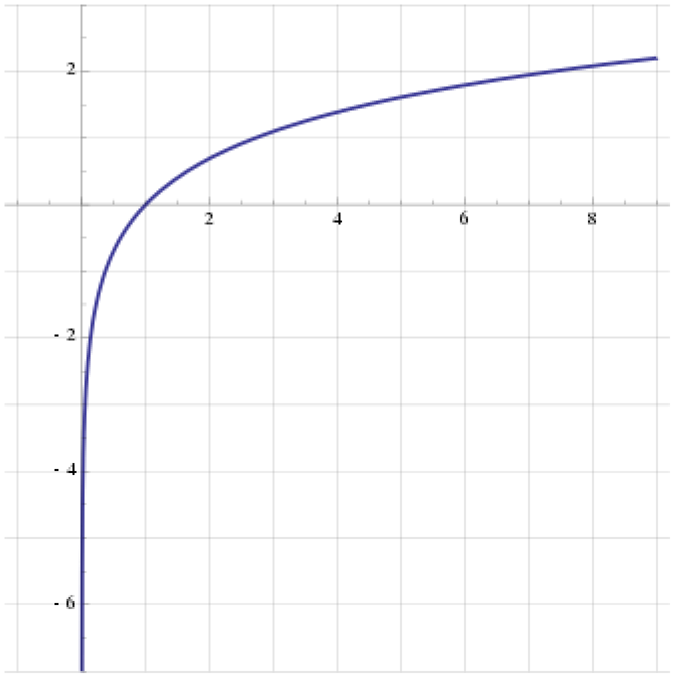
Cross entropy

- $\log \boldsymbol{softmax}(\boldsymbol{x})$
  - $\lim_{s \to 0^+} \log s = -\infty$



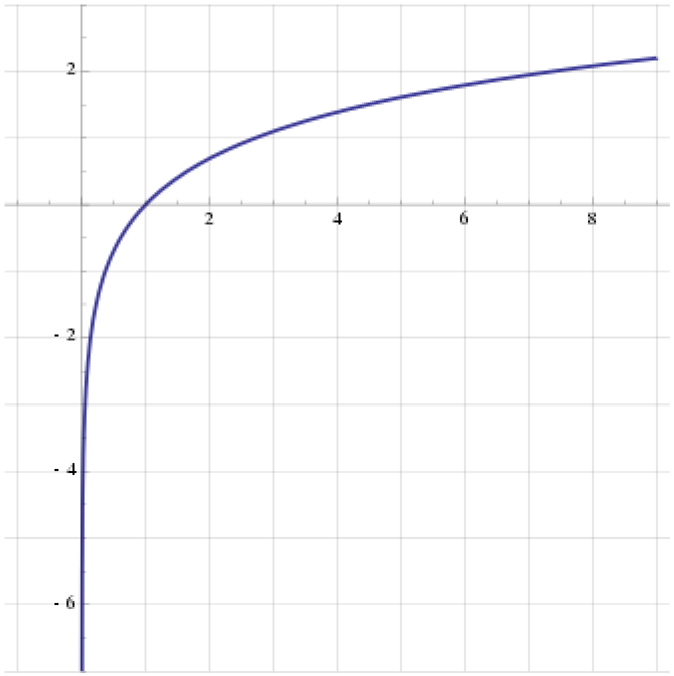
Cross entropy

- $\log softmax(\boldsymbol{x})$
  - $\lim_{s \to 0^+} \log s = -\infty$
- It will lead to NaN in the BP
  - $(\ln s)' = \frac{1}{s} \to \frac{1}{0} \to NaN$



Cross entropy

- $\log softmax(\boldsymbol{x})$
  - $\lim_{s \to 0^+} \log s = -\infty$
- It will lead to NaN in the BP
  - $(\ln s)' = \frac{1}{s} \to \frac{1}{0} \to NaN$
- Solution: set threshold
  - Given $\epsilon > 0$
  - $\log \max(\epsilon, softmax(\boldsymbol{x}))$
- Tensorflow
  - tf.nn.softmax_cross_entropy_with_logits



Poor conditioning

Matrix norm

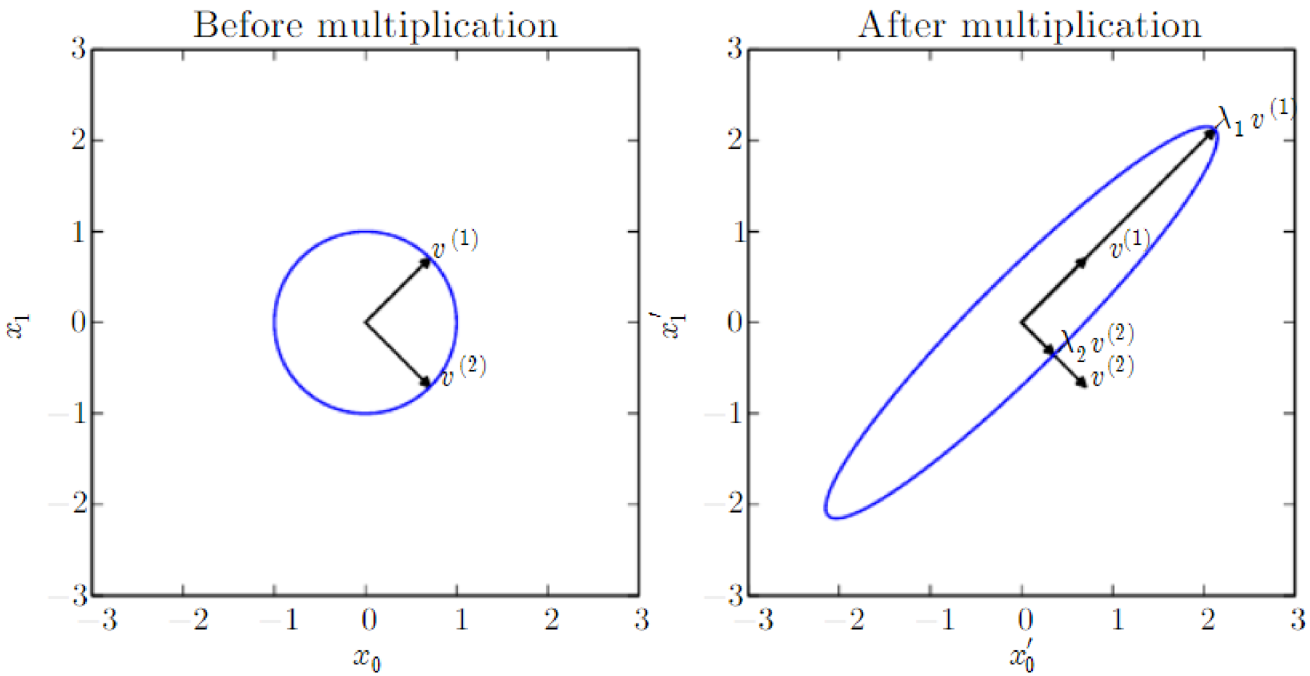
- $\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$
  - $\|\alpha A\| = |\alpha| \|A\|$
  - $\|A + B\| \leq \|A\| + \|B\|$
  - $\|A\| \geq 0$
  - $\|A\| = 0$ iff $A = 0$

- $\|AB\| \leq \|A\| \|B\|$

Matrix norm

- $\|A\|_p = \max\limits_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$
  - $\|\alpha A\| = |\alpha| \|A\|$
  - $\|A + B\| \leq \|A\| + \|B\|$
  - $\|A\| \geq 0$
  - $\|A\| = 0$ iff $A = 0$

- $\|AB\| \leq \|A\| \|B\|$



If A is square
Then $\|A\|_2 = \max\limits_i |\lambda_i|$

Relative errors

- In math, x
- In computer, $\bar{x}$
- Relative error:
  - $e_x = \frac{\|\bar{x} - x\|}{\|x\|}$

Relative errors

- In math, x
- In computer, $\bar{x}$
- Relative error:
  - $e_x = \frac{\|\bar{x}-x\|}{\|x\|} < ?$ ←we want to know

Relative errors

- In math, x
- In computer, $\bar{x}$
- Relative error:
  - $e_x = \frac{\|\bar{x}-x\|}{\|x\|} < ?$ ←we want to know
- How many digits can we believe in the fraction?
  - $-\log e_x$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y} - y\|}{\|y\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$

1. $A(\bar{y} - y) = n$
   $\bar{y} - y = A^{-1}n$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$

1. $A(\bar{y} - y) = n$

   $\bar{y} - y = A^{-1}n$

2. $\|x\| = \|Ay\|$

   $\|x\| \leq \|A\|\|y\|$

   $\frac{1}{\|y\|} \leq \frac{\|A\|}{\|x\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$
  - $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$

1. $A(\bar{y} - y) = n$
   $\bar{y} - y = A^{-1}n$
2. $\|x\| = \|Ay\|$
   $\|x\| \le \|A\|\|y\|$
   $\frac{1}{\|y\|} \le \frac{\|A\|}{\|x\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$
  - $e_y = \frac{\|\bar{y}-y\|}{\|y\|} = \frac{\|A^{-1}n\|}{\|y\|}$

1. $A(\bar{y} - y) = n$

   $\bar{y} - y = A^{-1}n$

2. $\|x\| = \|Ay\|$

   $\|x\| \leq \|A\|\|y\|$

   $\frac{1}{\|y\|} \leq \frac{\|A\|}{\|x\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$
  - $e_y = \frac{\|\bar{y}-y\|}{\|y\|} = \frac{\|A^{-1}n\|}{\|y\|} \leq \frac{\|A^{-1}\|\|n\|}{\|y\|}$

1. $A(\bar{y} - y) = n$
   $\bar{y} - y = A^{-1}n$

2. $\|x\| = \|Ay\|$
   $\|x\| \leq \|A\|\|y\|$
   $\frac{1}{\|y\|} \leq \frac{\|A\|}{\|x\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$
  - $e_y = \frac{\|\bar{y}-y\|}{\|y\|} = \frac{\|A^{-1}n\|}{\|y\|} \leq \frac{\|A^{-1}\|\|n\|}{\|y\|} \leq \frac{\|A\|\|A^{-1}\|\|n\|}{\|x\|}$

1. $A(\bar{y} - y) = n$
   $\bar{y} - y = A^{-1}n$
2. $\|x\| = \|Ay\|$
   $\|x\| \leq \|A\|\|y\|$
   $\frac{1}{\|y\|} \leq \frac{\|A\|}{\|x\|}$

Condition number

- $y = f(x) = A^{-1}x$
- Let $\bar{x} = x + n$, and then we have
  - $Ay = x$
  - $A\bar{y} = \bar{x} = x + n$
- Relative errors
  - $e_x = \frac{\|n\|}{\|x\|}$, $e_y = \frac{\|\bar{y}-y\|}{\|y\|}$

1. $A(\bar{y} - y) = n$
   $\bar{y} - y = A^{-1}n$

2. $\|x\| = \|Ay\|$
   $\|x\| \leq \|A\|\|y\|$
   $\frac{1}{\|y\|} \leq \frac{\|A\|}{\|x\|}$

- $e_y = \frac{\|\bar{y}-y\|}{\|y\|} = \frac{\|A^{-1}n\|}{\|y\|} \leq \frac{\|A^{-1}\|\|n\|}{\|y\|} \leq \frac{\|A\|\|A^{-1}\|\|n\|}{\|x\|} = \|A\|\|A^{-1}\|e_x$

Condition number cont.

- $y = f(x) = A^{-1}x$
  - $e_y \leq \|A\| \|A^{-1}\| e_x$
- Condition number
  - $\|A\| \|A^{-1}\| = \max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$
- Property
  - $\|A\| \|A^{-1}\| \geq \|AA^{-1}\| = \|I\| = 1$
  - y may have a larger error than x

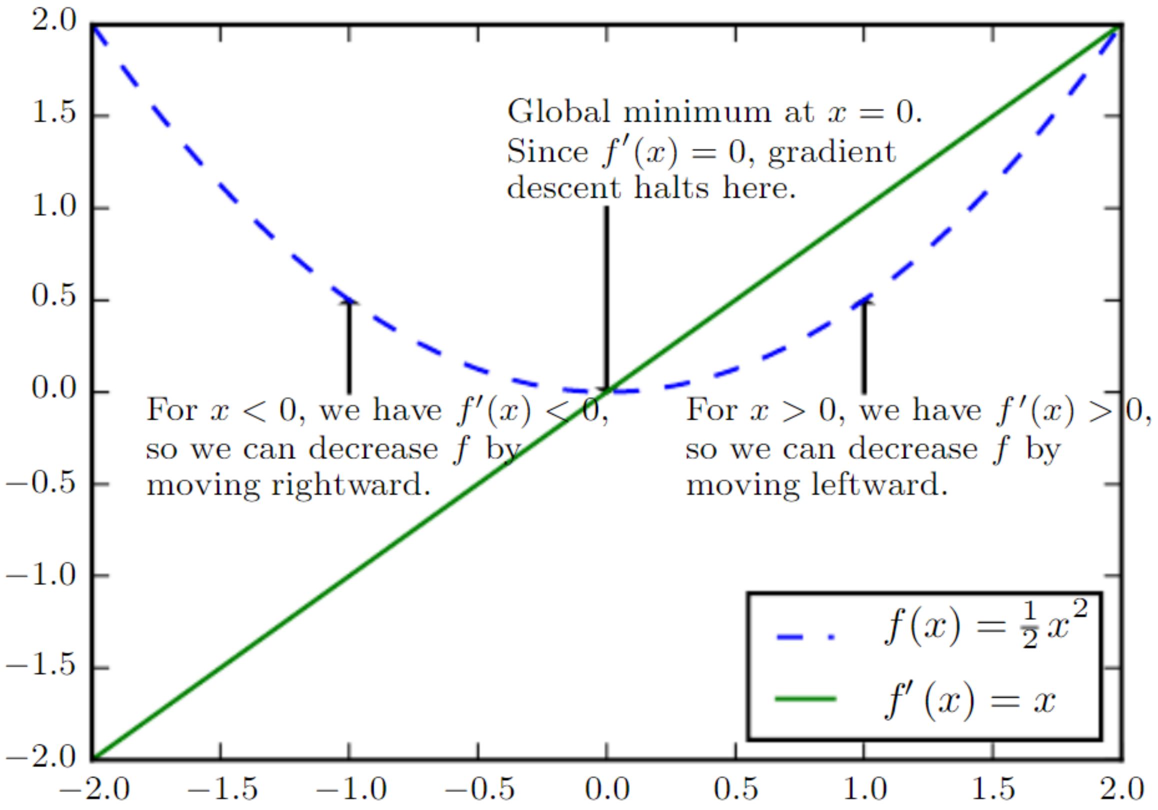
Gradient-based Optimization

Target

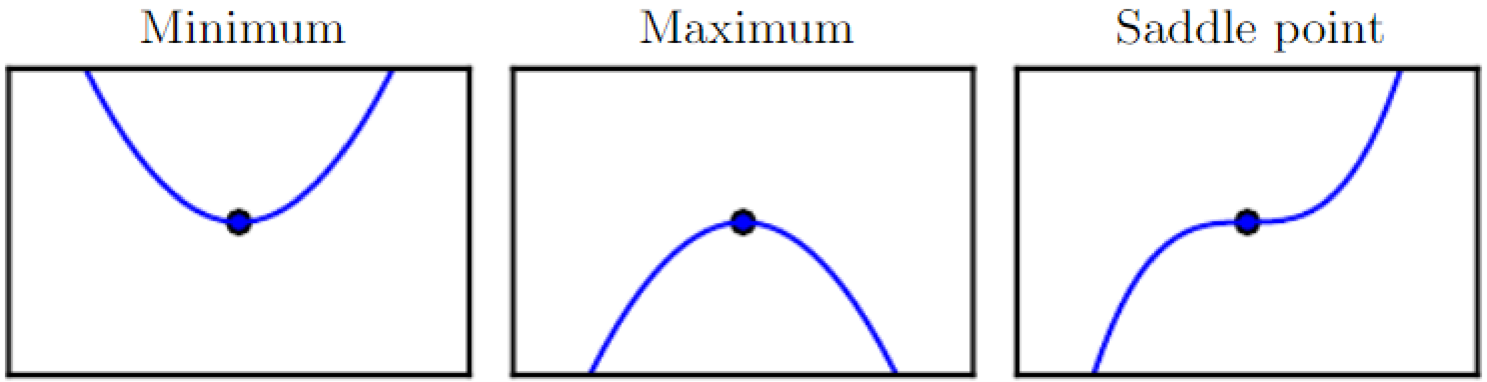
- Minimize or maximize some function $f(x)$
  - $\min f(x)$
  - $\max f(x) = \min -f(x)$
- $f(x)$
  - Objective function
  - Criterion
  - Cost function
  - Loss function
  - Error function
- $x^* = \operatorname{argmin} f(x)$

Gradient descent

- Given $\epsilon > 0$
- $f(x+\epsilon) \approx f(x) + \epsilon f'(x)$
- $f(x - \epsilon \operatorname{sign} f'(x))$



Critical point

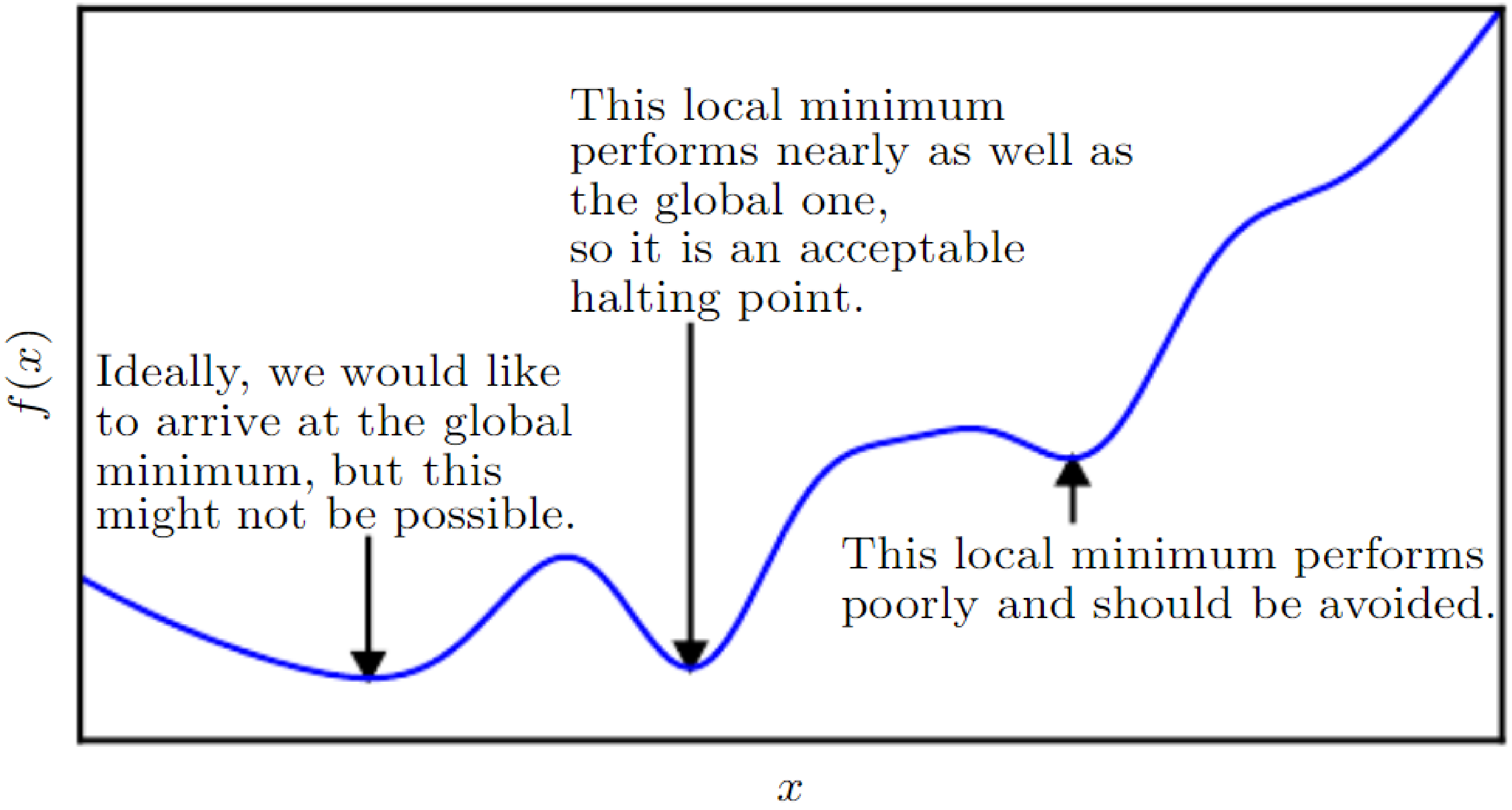


$f'(x) = 0$
$f''(x) > 0$

$f'(x) = 0$
$f''(x) < 0$

$f'(x) = 0$
$f''(x) = 0$

Approximate minimization



Partial derivatives

- Gradient
  - $\nabla_x f(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix}$

Directional derivative

- Given u with $\|u\| = 1$
- $D_u f(x) = \frac{\partial}{\partial \alpha} f(x + \alpha u) = \lim_{\alpha \to 0} \frac{f(x+\alpha u) - f(x)}{\alpha} = u^\top \nabla_x f(x)$

Directional derivative

- Given u with $\|u\| = 1$
- $D_u f(x) = \frac{\partial}{\partial \alpha} f(x + \alpha u) = \lim_{\alpha \to 0} \frac{f(x+\alpha u) - f(x)}{\alpha} = u^\top \nabla_x f(x)$
- Find u such that

$$\min_{u, u^\top u = 1} u^\top \nabla_x f(x)$$

$$= \min_{u, u^\top u = 1} \|u^\top\|_2 \, \|\nabla_x f(x)\|_2 \cos\theta$$

$$= \min_{u, u^\top u = 1} \cos\theta$$

Optimization

Steepest descent

- $x' = x - \epsilon \nabla_x f(x)$
  - ϵ: learning rate
  - Named gradient descent, too.

Optimization

Steepest descent

- $x' = x - \epsilon \nabla_x f(x)$
  - ϵ: learning rate
  - Named gradient descent, too.

Line search

- Base on steepest descent
- Find the best ϵ
  - $\min_{\epsilon} f\left(x - \epsilon \nabla_x f(x)\right)$

Something else

Steepest descent

- We can solve the equation $\nabla_x f(x) = 0$ for x directly
  - w/o iteration i.e., $x' = x - \epsilon \nabla_x f(x)$

Hill climbing

- Ascending an objective function of discrete parameters

Jacobian matrix

- $f: \mathbb{R}^m \to \mathbb{R}^n$
- $J \in \mathbb{R}^{n \times m}$ of f is defined $s.t.$

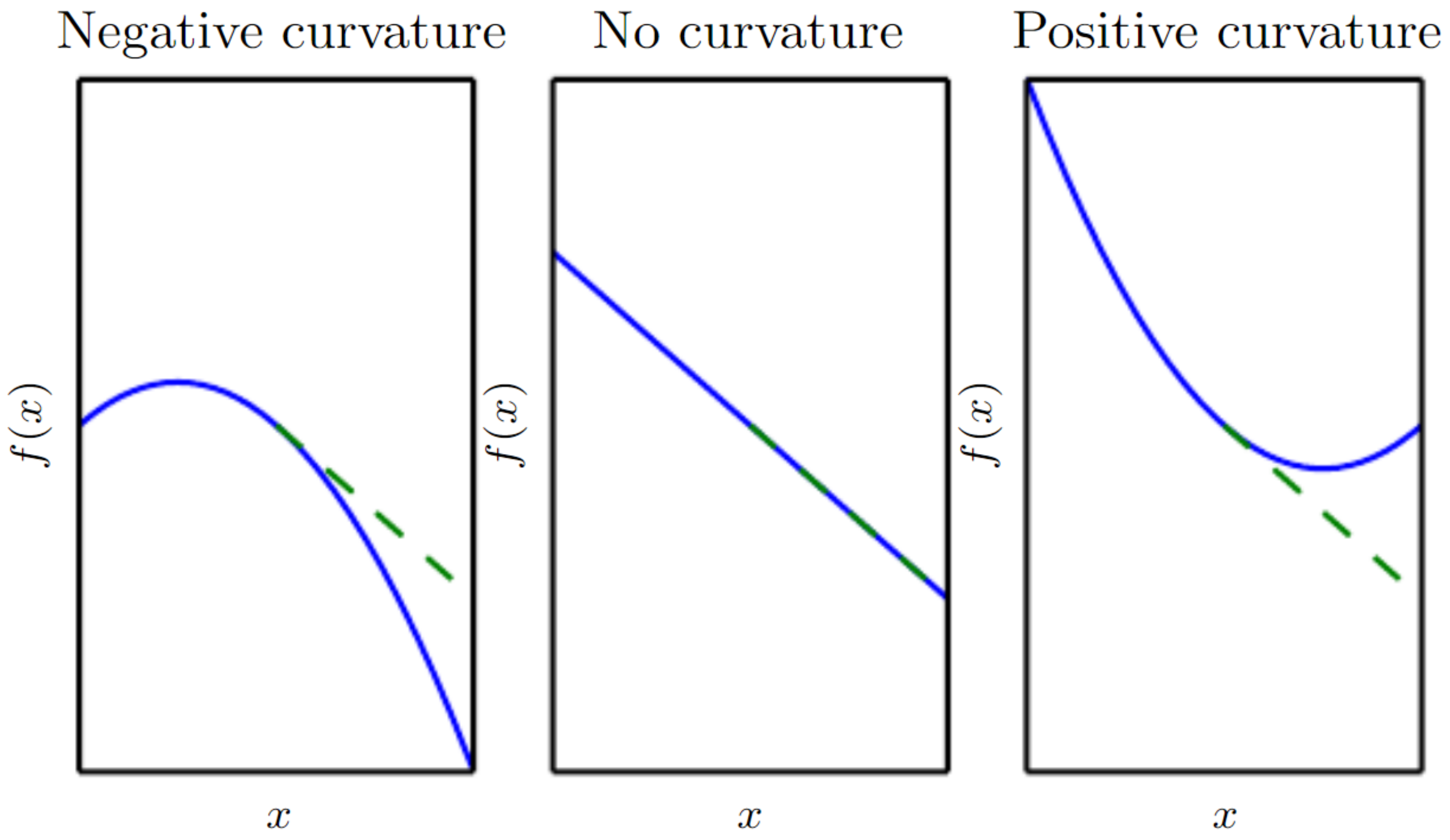
$$J_{i,j} = \frac{\partial}{\partial x_j} f(x)_i$$

- That is

$$J = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x)_1 & \cdots & \frac{\partial}{\partial x_m} f(x)_1 \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f(x)_n & \cdots & \frac{\partial}{\partial x_m} f(x)_n \end{bmatrix}$$

Curvature

- In a single dimension
- $f''(x) = \frac{d^2}{dx^2} f$



$f''(x) < 0$ $f''(x) = 0$ $f''(x) > 0$

Hessian matrix

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

- If the second partial derivatives are continuous
  - Then $\frac{\partial^2}{\partial x_i \partial x_j} f(x) = \frac{\partial^2}{\partial x_j \partial x_i} f(x)$
  - Imply that H is symmetric

Hessian matrix

- Given a unit vector d
- The second derivative is $d^\top H d$

Recall: directional derivative
$u^\top \nabla_x f(x)$

Hessian matrix

- Given a unit vector d
- The second derivative is $d^\top H d$
- When d is eigenvector of H
  - The value of 2nd derivative equals its eigenvalue λ
  - Maximum eigenvalue → maximum 2nd derivative
  - Minimum eigenvalue → minimum 2nd derivative

Hessian matrix

- When g is the gradient and H is the Hessian at $x^{(0)}$
  - Second-order Taylor series

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

- Let $x = x^{(0)} - \epsilon g$ (i.e., gradient descent)

$$f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^{\top} g + \frac{1}{2}\epsilon^{2} g^{\top} H g$$

Hessian matrix for gradient descent

- $f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^\top g + \frac{1}{2}\epsilon^2 g^\top H g$

Hessian matrix for gradient descent

- $f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^\top g + \frac{1}{2}\epsilon^2 g^\top H g$
- If $g^\top H g$ is too large
  - Then GD could move uphill

Hessian matrix for gradient descent

- $f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^\top g + \frac{1}{2}\epsilon^2 g^\top H g$
- If $g^\top H g$ is too large
  - Then GD could move uphill
- If $g^\top H g$ is zero or negative
  - Then GD will decrease f forever

Hessian matrix for gradient descent

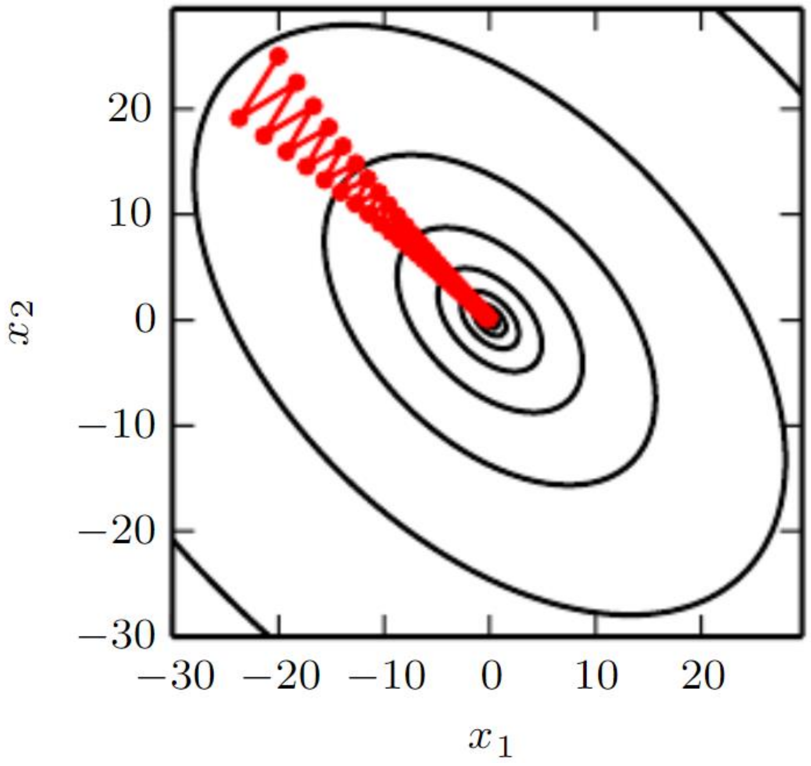
- $f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^{\top} g + \frac{1}{2}\epsilon^2 g^{\top} H g$
- If $g^{\top} H g$ is too large
  - Then GD could move uphill
- If $g^{\top} H g$ is zero or negative
  - Then GD will decrease f forever
- When $g^{\top} H g$ is positive
  - Choose $\epsilon^* = \frac{g^{\top} g}{g^{\top} H g}$
  - Then $-\epsilon g^{\top} g + \frac{1}{2}\epsilon^2 g^{\top} H g = -\frac{\left(g^{\top} g\right)^2}{g^{\top} H g} + \frac{1}{2}\frac{\left(g^{\top} g\right)^2}{g^{\top} H g} = -\frac{1}{2}\frac{\left(g^{\top} g\right)^2}{g^{\top} H g} < 0$

Hessian matrix for gradient descent

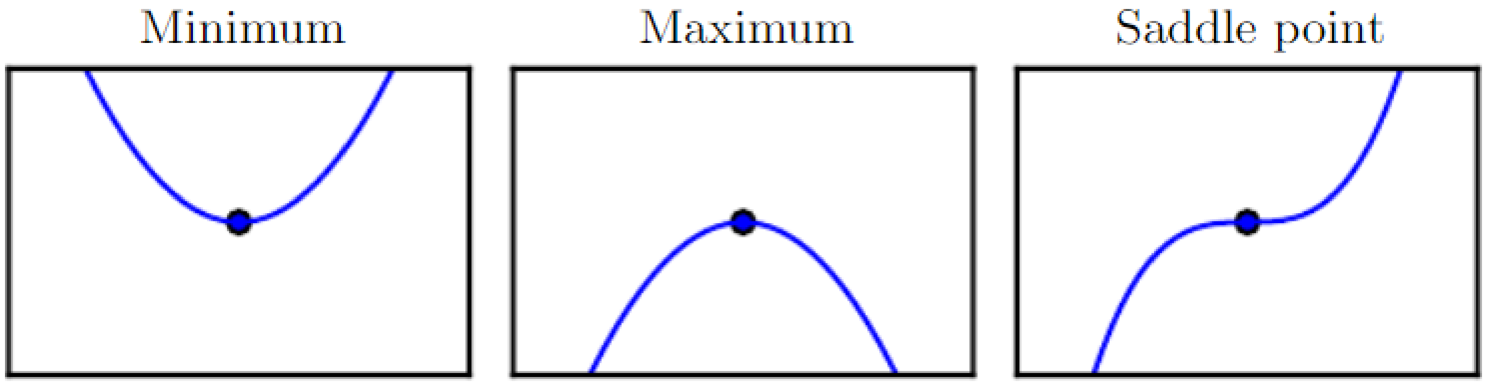
- $f\left(x^{(0)} - \epsilon g\right) \approx f\left(x^{(0)}\right) - \epsilon g^\top g + \frac{1}{2}\epsilon^2 g^\top H g$
- When $g^\top H g$ is positive
  - Choose $\epsilon^* = \frac{g^\top g}{g^\top H g}$
- In worst case, g is eigenvector of H with maximal eigenvalue
  - $\epsilon^* = \frac{1}{\lambda_{max}}$

Hessian matrix

- Condition number $= 5$



Recall: Critical point



$f'(x) = 0$ $f''(x) > 0$	$f'(x) = 0$ $f''(x) < 0$	$f'(x) = 0$ $f''(x) = 0$

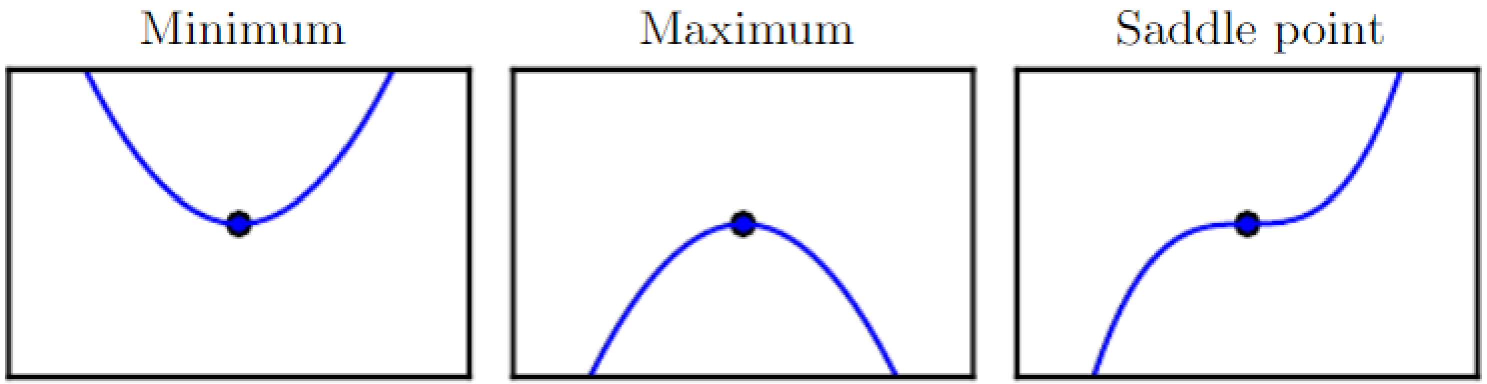
2^{nd} derivative test

- When $f'(x) = 0$, check $f''(x)$

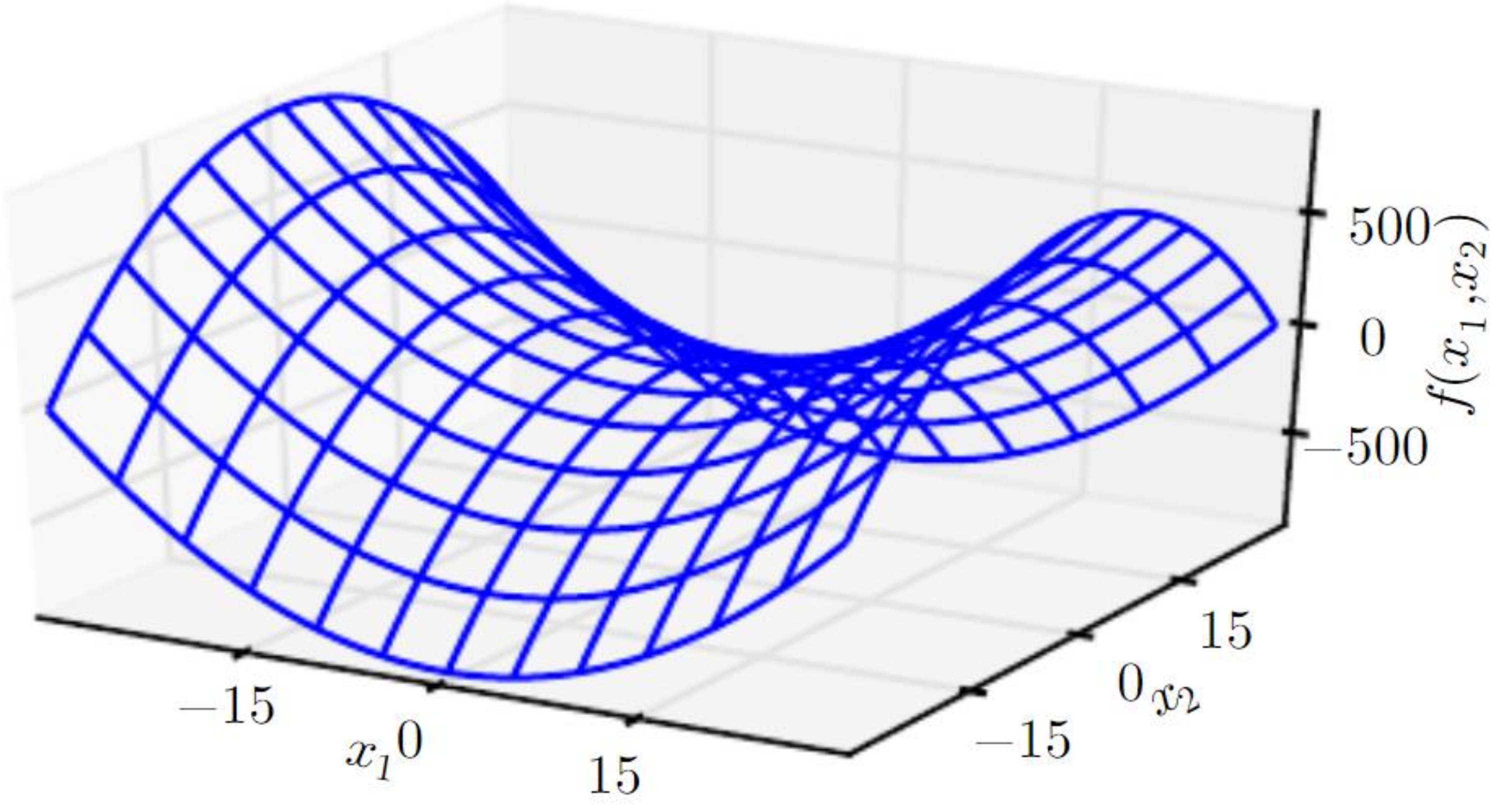
  If $f''(x) > 0$

  - $f'(x - \epsilon) < 0$ and $f'(x + \epsilon) > 0$
  - Local minimum

  If … (be omitted)



Saddle point



Newton’s method

- Find root of $f(x)$ i.e., $f(x) = 0$
  - $x^* = x - \frac{f(x)}{f'(x)}$

Newton's method

- Find root of $f(x)$ i.e., $f(x) = 0$
  - $x^* = x - \frac{f(x)}{f'(x)}$
- Find critical point i.e., $f'(x) = 0$
  - $x^* = x - \frac{f'(x)}{f''(x)}$

Newton’s method

- Find root of $f(x)$ i.e., $f(x) = 0$
  - $x^* = x - \frac{f(x)}{f'(x)}$
- Find critical point i.e., $f'(x) = 0$
  - $x^* = x - \frac{f'(x)}{f''(x)}$
- In multi-dimension
  - $x^* = x^{(0)} - \left(H(f)\left(x^{(0)}\right)\right)^{-1} \nabla_x f\left(x^{(0)}\right)$

Efficiency

- First-order optimization algorithms
  - Gradient descent
  - Bisection method
- second-order optimization algorithms
  - Newton’s method

- Lipschitz continue
  - $\forall x, \forall y, |f(x) - f(y)| \leq \mathcal{L}\|x - y\|_2$
- Convex optimization
  - Lack saddle points
  - All their local minima are necessarily global minima
  - However, most problems in deep learning are difficult to express in terms of convex optimization

Under construction

- Lipschitz continue
  - $\forall x, \forall y, |f(x) - f(y)| \leq \mathcal{L}||x - y||_2$
- Convex optimization
  - Lack saddle points
  - All their local minima are necessarily global minima
  - However, most problems in deep learning are difficult to express in terms of convex optimization

Constrained Optimization

Karush-Kuhn-Tucker approach

- Let $\mathbb{S} = \left\{x | \forall i, g^{(i)}(x) = 0 \text{ and } \forall j, h^{(j)}(x) \leq 0\right\}$
- $\min f(x)$ subject to $x \in \mathbb{S}$

$$L(x, \lambda, \alpha) = f(x) + \sum_i \lambda_i \, g^{(i)}(x) + \sum_j \alpha_j \, g^{(j)}(x)$$

Karush-Kuhn-Tucker approach

- Let $\mathbb{S} = \left\{x | \forall i, g^{(i)}(x) = 0 \text{ and } \forall j, h^{(j)}(x) \leq 0\right\}$
- $\min f(x)$ subject to $x \in \mathbb{S}$

$$L(x, \lambda, \alpha) = f(x) + \sum_i \lambda_i \, g^{(i)}(x) + \sum_j \alpha_j \, g^{(j)}(x)$$

- Now $\min_{x \in \mathbb{S}} f(x) = \min_x \max_\lambda \max_{\alpha, \alpha \geq 0} L(x, \lambda, \alpha)$

Karush-Kuhn-Tucker approach

- Let $\mathbb{S} = \{x | \forall i, g^{(i)}(x) = 0 \text{ and } \forall j, h^{(j)}(x) \leq 0\}$
- $\min f(x)$ subject to $x \in \mathbb{S}$

$$L(x, \lambda, \alpha) = f(x) + \sum_i \lambda_i \, g^{(i)}(x) + \sum_j \alpha_j \, g^{(j)}(x)$$

- Now $\min_{x \in \mathbb{S}} f(x) = \min_x \max_{\lambda} \max_{\alpha, \alpha \geq 0} L(x, \lambda, \alpha)$
  - Satisfied: $\max_{\lambda} \max_{\alpha, \alpha \geq 0} L(x, \lambda, \alpha) = f(x)$
  - Violated: $\max_{\lambda} \max_{\alpha, \alpha \geq 0} L(x, \lambda, \alpha) = \infty$

Example: Linear Least Squares

Linear least squares

- Find the value of x that minimizes
  - $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

Gradient descent

- Find the value of x that minimizes
  - $f(x) = \frac{1}{2}\|Ax - b\|_2^2$
- First, gradient descent
  - $\nabla_x f(x) = A^\top(Ax - b) = A^\top Ax - A^\top b$
  - $x \leftarrow x - \epsilon(A^\top Ax - A^\top b)$

Gradient descent

- Find the value of x that minimizes
  - $f(x) = \frac{1}{2}||Ax - b||_2^2$
- First, gradient descent
  - $\nabla_x f(x) = A^\top(Ax - b) = A^\top Ax - A^\top b$
  - $x \leftarrow x - \epsilon(A^\top Ax - A^\top b)$

Algorithm 4.1 An algorithm to minimize $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$ with respect to $\boldsymbol{x}$ using gradient descent, starting from an arbitrary value of $\boldsymbol{x}$.

Set the step size (ϵ) and tolerance (δ) to small, positive numbers.
while $||\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}||_2 > \delta$ **do**
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \epsilon\left(\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}\right)$
end while

Newton’s method

- For simple example
  - $f(x) = \frac{1}{2}(3x-2)^2 + 5 \ \rightarrow \ f'(x) = 9x - 6 \ \rightarrow f''(x) = 9$
  - In junior high school, we know $x^* = \frac{2}{3}$

Newton's method

- For simple example
  - $f(x) = \frac{1}{2}(3x-2)^2 + 5 \;\rightarrow\; f'(x) = 9x - 6 \;\rightarrow f''(x) = 9$
  - In junior high school, we know $x^* = \frac{2}{3}$
- At $x = 4$
  - $f'(4) = 9 \times 4 - 6 = 30$
  - $f''(4) = 9$
  - $x^* = x - \frac{f'(x)}{f''(x)} = 4 - \frac{30}{9} = 4 - \frac{10}{3} = \frac{2}{3}$
- Only one step!!!!

Karush-Kuhn-Tucker

- Find the value of x that minimizes
  - $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ subjects to $x^\top x \leq 1$
- Lagrangian
  - $L(x, \lambda) = f(x) + \lambda(x^\top x - 1)$
- We can now solve the problem
  - $\min_{x} \max_{\lambda, \lambda \geq 0} L(x, \lambda)$

$$\frac{\partial}{\partial x} L(x, \lambda) = A^\top A x - A^\top b + 2\lambda x = 0$$

$$x = (A^\top A x + 2\lambda I)^{-1} A^\top b$$

i.e. Moore-Penrose pseudoinverse

$$\frac{\partial}{\partial \lambda} L(x, \lambda) = x^\top x - 1$$

Thanks