# Numerical Computation

Vishnu Lokhande

Department of Computer Science and Engineering
University at Buffalo, SUNY
vishnulo@buffalo.edu

February 3rd, 2025

## Overflow and Underflow

1. Overflow happens when a number gets too large; While underflow happens when a number gets too small.
2. The exponentiation can underflow when the argument is very negative, or overflow when it is very positive.

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

3. How to deal with this?

# Overflow and Underflow

1. Overflow happens when a number gets too large; While underflow happens when a number gets too small.
2. The exponentiation can underflow when the argument is very negative, or overflow when it is very positive.

$$\mathrm{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

3. How to deal with this?

$$\mathrm{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}} = \frac{e^{x_i}/e^{\max x_{i'}}}{\sum_j e^{x_j}/e^{\max x_{i'}}} \\ = \frac{e^{x_i - \max x_{i'}}}{\sum_j e^{x_j - \max x_{i'}}}$$

   - Potentially gets underflow, but usually not a problem in practice because we care about the largest value.

# Condition Number

1. Conditioning refers to how rapidly a function changes with a small change in input.
2. Consider $f(\mathbf{x}) = \mathbf{A}^{-1}\,\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ has a engendecomposition with eigenvalues $\{\lambda_i\}$.
    - The condition number of **A** is defined as $\max_{i,j} \left|\frac{\lambda_i}{\lambda_j}\right|$.
    - When this is large, the output $f(\mathbf{x})$ is very sensitive to input error (perturbation), *i.e.*, the inversion is inaccurate.
    - Poorly conditioned matrices amplify pre-existing errors when we multiply by its inverse.

Why?

# Condition Number

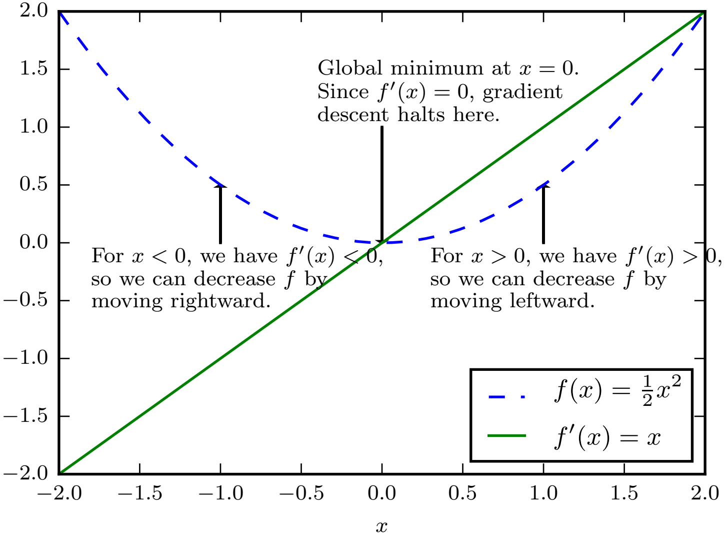
1. Conditioning refers to how rapidly a function changes with a small change in input.
2. Consider $f(\mathbf{x}) = \mathbf{A}^{-1}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ has a engendecomposition with eigenvalues $\{\lambda_i\}$.
   - The condition number of **A** is defined as $\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$.
   - When this is large, the output $f(\mathbf{x})$ is very sensitive to input error (perturbation), *i.e.*, the inversion is inaccurate.
   - Poorly conditioned matrices amplify pre-existing errors when we multiply by its inverse.

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T = \sum_i \lambda_i \underbrace{\mathbf{Q}_{:,i}\mathbf{Q}_{:,i}^T}_{\text{basis}}$$

$$\mathbf{A}^{-1}\delta = \sum_i \frac{\delta}{\lambda_i} \mathbf{Q}_{:,i}\mathbf{Q}_{:,i}^T$$

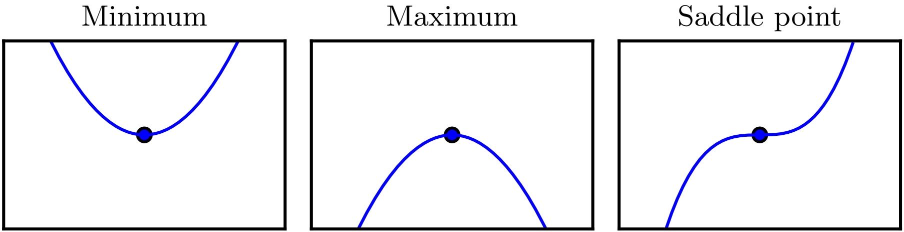
# Gradient Descent

## How to find the minimum value of a function $f$?

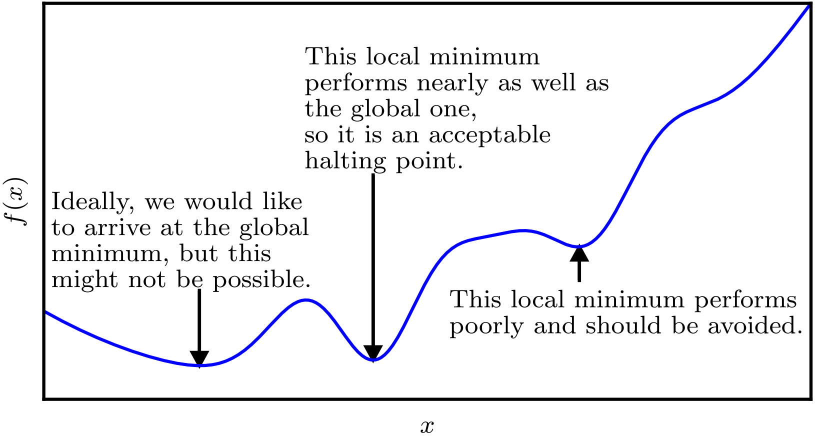


# Critical Points

### Gradient descent not always works.



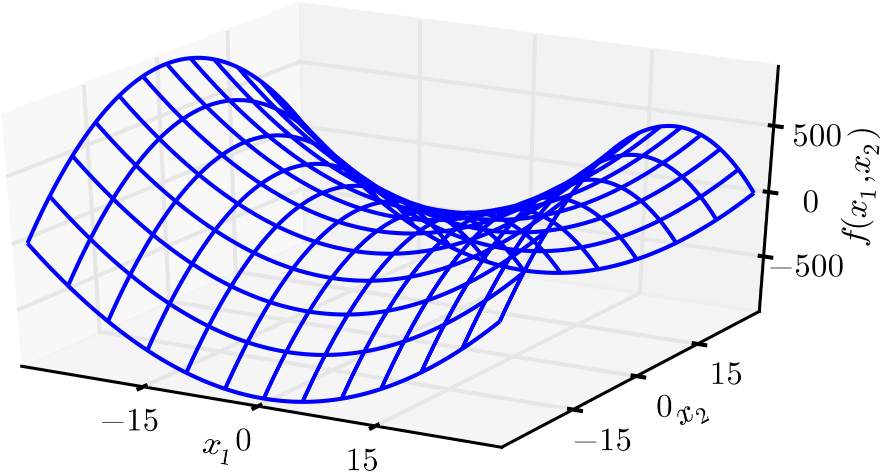
# Approximate Optimization



1. In deep neural networks, it is found local optima is typically not a problem:
   - All local optima are global optima under some conditions.

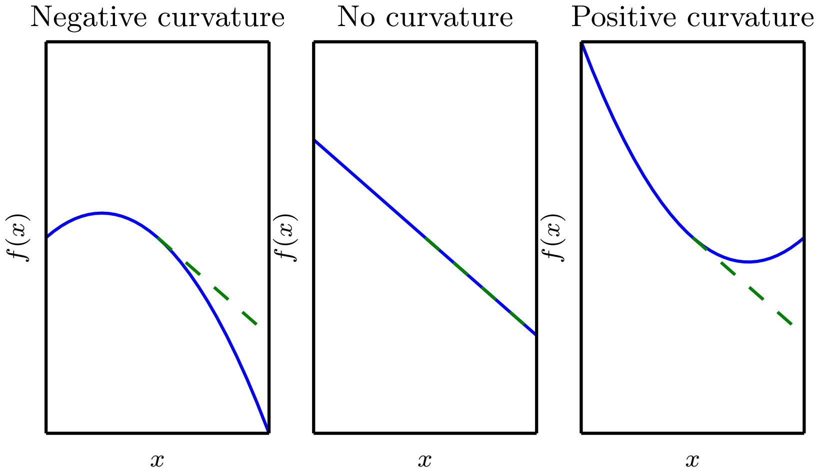
Kawaguchi, NIPS 2016

# Saddle Points



## Curvature

1. Local minimum/maximum has positive/negative curvature in all directions.
2. Saddle points have both positive and negative curvature.



# Hessian

1. Second derivatives of the objective function $f(\mathbf{x})$:

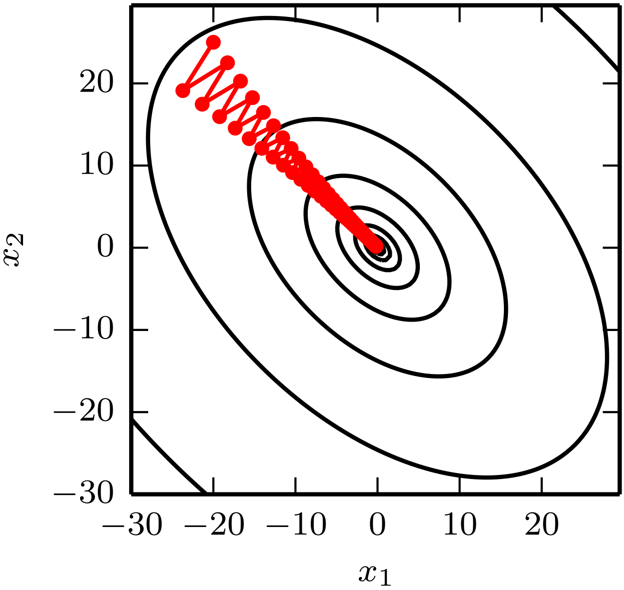
$$\mathbf{H}_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x})$$

2. Hessian is the Jacobian of the gradient.
3. The Hessian matrix is symmetric, *i.e.*, $\mathbf{H}_{ij} = \mathbf{H}_{ji}$.
   - It can be decomposed into a set of real eigenvalues and an orthogonal basis of eigenvectors: $\mathbf{H} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^T$.

# Pooring Conditioning

1. There are different second derivatives in each direction at a single point.
2. Condition number of **H**, *e.g.*, $\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$ measures how much they differ:
    - Gradient descent performs poorly when **H** has a poor condition number, because dirivatives in different dimensions are uneven.
    - Step size must be small so as to avoid overshooting the minimum, but it will be too small to make progress in other directions with less curvature.

# Gradient Descent with Poor Conditioning



# Constrained Optimization: Example (Optional)

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2} \|\mathbf{A}\,\mathbf{x} - \mathbf{b}\|^2$$
$$\text{s.t. } \mathbf{x}^T \mathbf{x} \leq 1$$

- Introduce the Lagrangian $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda \left(\mathbf{x}^T \mathbf{x} - 1\right)$, and solve

$$\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$$

- When $\frac{\partial L(\mathbf{x},\lambda)}{\partial \mathbf{x}}|_{\mathbf{x}^*} = 0$ and $\lambda \left(\mathbf{x}^{*T} \mathbf{x}^* - 1\right) \leq 0$, then $\mathbf{x}_i^*$ is an optimal solution for the original problem $\min_{\mathbf{x}} f(\mathbf{x})$.
- The conditions $\frac{\partial L(\mathbf{x},\lambda)}{\partial \mathbf{x}}|_{\mathbf{x}^*} = 0$ and $\lambda \left(\mathbf{x}^{*T} \mathbf{x}^* - 1\right) \leq 0$ are called the Karush-Kuhn-Tucker conditions (KKT).

# Constrained Optimization: KKT Conditions (Optional)

$$\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda \left( \mathbf{x}^T \mathbf{x} - 1 \right)$$

1. If $\mathbf{x}^T \mathbf{x} - 1 \leq 0$, to maximize w.r.t. $\lambda$, $\lambda$ needs to be set to 0:
   - recover the original problem
2. If $\mathbf{x}^T \mathbf{x} - 1 > 0$, to maximize w.r.t. $\lambda$, $\lambda = \infty$; However, the $\min_{\mathbf{x}}$ part will change the value of **x** to avoid $L$ to be infinity, until the min and max reach a balance, *i.e.*, the conditions $\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} |_{\mathbf{x}^*} = 0$ and $\lambda \left( \mathbf{x}^{*T} \mathbf{x}^* - 1 \right) \leq 0$ satisfy.
3. In practice, we can use gradient descent to approximately solve for **x** and $\lambda$ alternatively.