Northeastern University

Optimization

Roi Yehoshua

Agenda

Northeastern University

- Optimization problems
- Analytical methods
- Convex functions
- Iterative optimization methods
- Convergence rates
- Gradient descent
- Newton's method
- Quasi-Newton methods
- Constrained optimization

Optimization Problems

Northeastern University

- Optimization is the minimization or maximization of a function
  - possible subject to constraints on the variables
- We are given a scalar-valued **objective function** $f(\mathbf{x})$: $R^n \rightarrow R$
  - **x** is the vector of **variables** (also called unknowns or parameters)
- Two types of optimization problems:
  - **Unconstrained optimization:** there is no restriction on the values of the variables:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

  - **Constrained optimization** include explicit constraints on the variables:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, k \\ & h_j(\mathbf{x}) = 0 \quad j = 1, \ldots, p \end{aligned}$$

Local vs. Global Optima

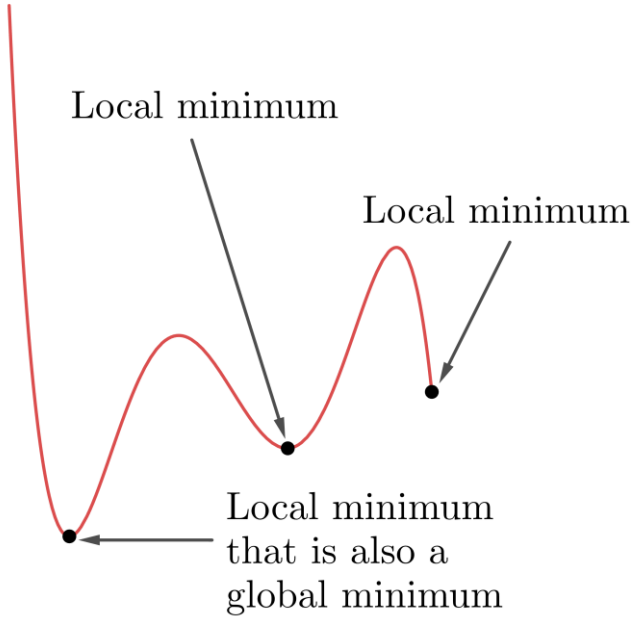
Northeastern University

- A point *x** is a **global minimizer** if

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \text{for all } \mathbf{x}$$

- A point x* is a **local minimizer** if there is a neighborhood *N* of x* such that

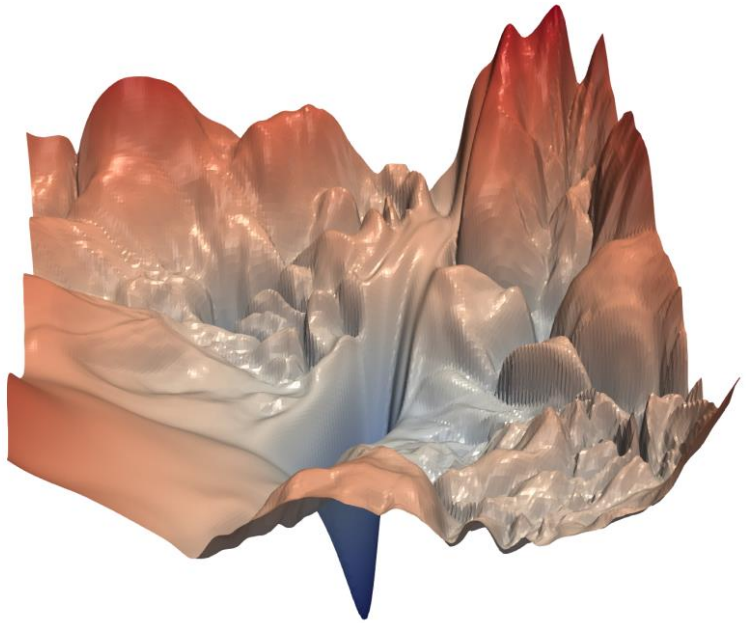
$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathcal{N}$$



Objective Surface

Northeastern University

- Many objective/loss functions in machine learning have complex surfaces
  - With many local minima and saddle points
  - Makes optimization difficult as algorithms tend to get “trapped” in local minima
- Example for a loss landscape of a neural network



Li et al., “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 2018

Optimization Methods

Northeastern University

- **Analytical methods**
  - Using calculus, usually yields a closed-form solution
- **Iterative methods**
  - **First-order methods**
    - Based on the gradient vector
    - Gradient descent and its variants
    - Coordinate descent
    - Adaptive methods used in deep learning (RMSProp, AdaDelta, Adam)
  - **Second-order methods**
    - Newton's method
    - Quasi-newton methods (e.g., BFGS)
- **Heuristic methods**
  - Genetic algorithms, simulated annealing, particle swarm optimization

Analytical Methods

Northeastern University

- Find exact solutions to the optimization problem
- Typically, can be applied only to small-scale problems with simple objective functions
- Consist of two main steps:
  - Identify the **critical points** of the objective function, where the gradient vanishes

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

  - Use the second-order derivative (Hessian) $H_f(\mathbf{x})$ to classify the critical points:

Definiteness of *H*	Eigenvalues of *H*	Type of critical point
Positive definite	All positive	local minimum
Negative definite	All negative	local maximum
Indefinite	Both positive and negative	saddle point
Semidefinite (positive or negative)		test is inconclusive

Example

Northeastern University

- Find the critical points of the following function

$$f(x, y) = x^2 - y^2$$

- We first compute the gradient of the function:

$$\nabla f = \begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ -2y \end{pmatrix}$$

- Setting the gradient to zero, we obtain the system:

$$2x = 0, \quad -2y = 0$$

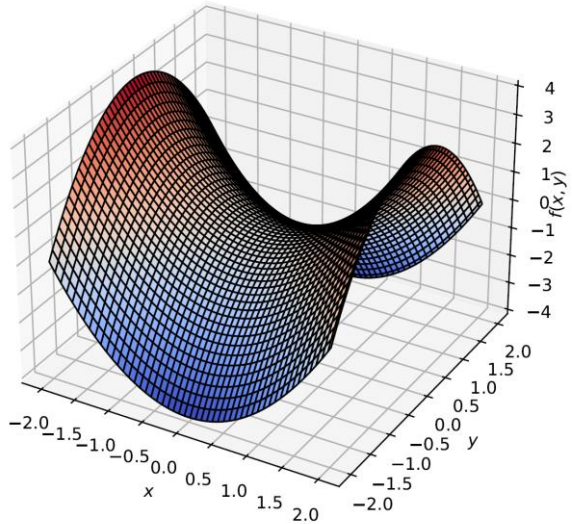
  - Thus (0, 0) is the only critical point

Example

- Computing the Hessian matrix:

$$H_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

  - Since it is a diagonal matrix, its eigenvalues are the diagonal entries: 2 and -2
  - Thus, the Hessian is indefinite (it has both positive and negative eigenvalues)
  - Conclusion: (0, 0) is a **saddle point**



Sylvester's Criterion

Northeastern University

- Computing the eigenvalues of the Hessian can be time-consuming
- Instead, we can determine the definiteness of *H* using its **leading principal minors**

$$D_k = \det(H_{1:k,\,1:k})$$

- **Sylvester's criterion** states that:
  - *H* is positive definite if all leading principal minors are positive:

$$D_1 > 0, \quad D_2 > 0, \quad D_3 > 0, \quad D_4 > 0, \quad \ldots$$

  - H is negative definite if the leading principal minors alternate in sign:

$$D_1 < 0, \quad D_2 > 0, \quad D_3 < 0, \quad D_4 > 0, \quad \ldots$$

  - If neither condition holds, *H* can be indefinite or semidefinite and further analysis needed

Sylvester's Criterion

Northeastern University

- For a function $f(x, y)$ of two variables, the Hessian is:

$$H_f = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$$

- The two leading principal minors are:

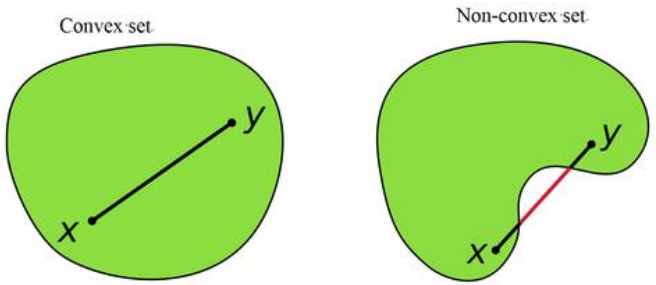
$$D_1 = f_{xx}, \quad D_2 = \det(H_f) = f_{xx} f_{yy} - (f_{xy})^2$$

- By Sylvester's criterion:
  - If $\det(H_f) > 0$ and $f_{xx} > 0$, H_f is positive definite and the critical point is local minimum
  - If $\det(H_f) > 0$ and $f_{xx} < 0$, H_f is negative definite and the critical point is local maximum
  - If $\det(H_f) < 0$, H_f is indefinite and the critical point is a saddle point
  - If $\det(H_f) = 0$, the test is inconclusive

Convex Sets

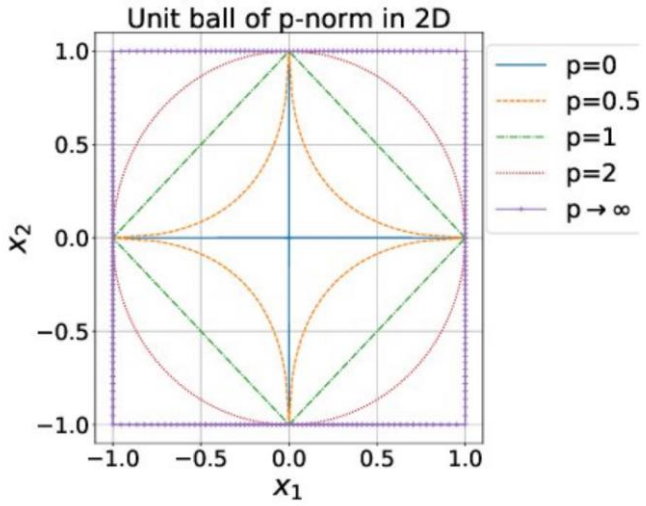
- A set $S \subseteq R^n$ is **convex** if the line connecting any two points in S lies entirely within S
  - For any two points $\mathbf{x}_1, \mathbf{x}_2 \in S$ and any $\lambda \in [0, 1]$

$$\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in \mathcal{S}$$



- Examples for convex sets:
  - The entire Euclidean space R^n
  - Any interval in R, such as $[a, b]$, $[a, b)$, or (a, b)
  - Norm balls are convex for any norm

$$B_r(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{x}_0\| \leq r\}$$

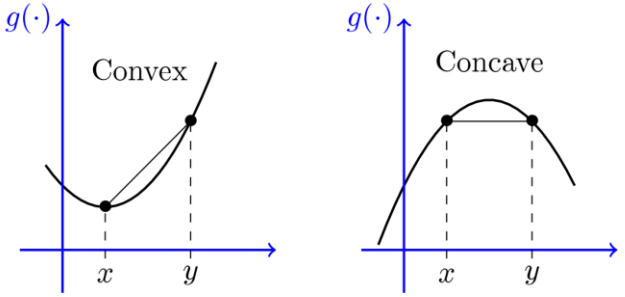


Convex Functions

Northeastern University

- $f: S \rightarrow R$ is a **convex function** if its domain S is a convex set and for any two points $\mathbf{x}_1$, $\mathbf{x}_2$ in S, the graph of f lies below the line connecting $(\mathbf{x}_1, f(\mathbf{x}_1))$ and $(\mathbf{x}_2, f(\mathbf{x}_2))$
  - For any two points $\mathbf{x}_1, \mathbf{x}_2 \in S$ and any $\lambda \in [0, 1]$

$$f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$$



- f is a **concave function** if $-f$ is convex
- f is **strictly convex** if the inequality is strict whenever $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\lambda \in (0, 1)$

$$f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) < \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$$

- Examples:
  - Linear functions $f(x) = ax + b$ are both convex and concave (but not strict)
  - Quadratic functions $f(x) = ax^2 + bx + c$ are strictly convex if $a > 0$ and strictly concave if $a < 0$
  - Exponential function $f(x) = e^x$ is strictly convex

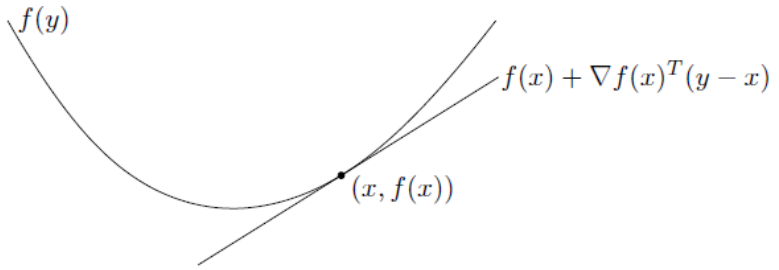
Conditions for Convexity

Northeastern University

- Let f: $S \rightarrow R$ be a function defined on a convex set S
- **First-order condition**: if f is differentiable, then it is convex if and only if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla_{\mathbf{x}} f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}), \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{S}$$

  - The graph of f lies above or on its tangent plane



- **Second-order condition**: if f is twice differentiable, then it is convex if and only if its Hessian H_f is positive semidefinite at every point in its domain

$$H_f(\mathbf{x}) \succeq 0, \quad \text{for all } \mathbf{x} \in \mathcal{S}$$

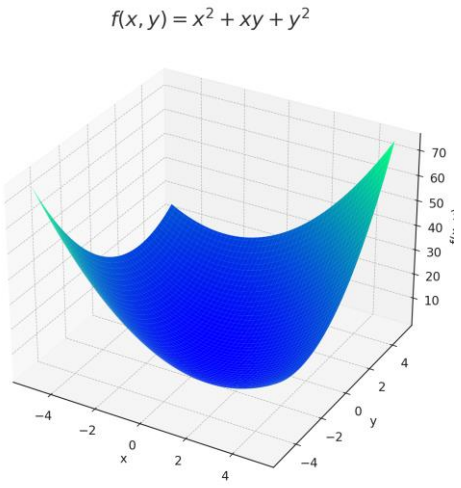
  - If H_f is positive definite, then the function is strictly convex

Conditions for Convexity

- Example: Prove that the function $f(x, y) = x^2 + xy + y^2$ is convex
- The Hessian of the function is constant:

$$H_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

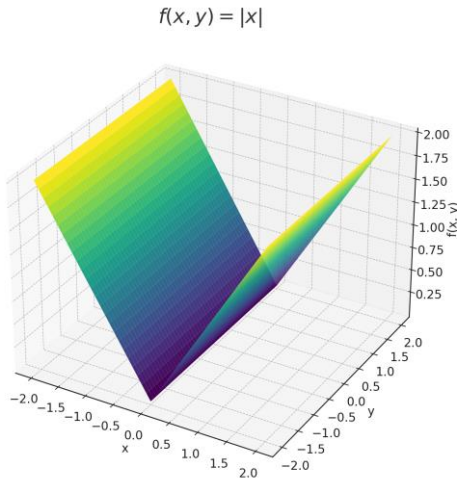
- To determine whether H_f is positive definite, we use Sylvester's criterion:
  - The first leading principal minor is $2 > 0$
  - The second leading principal minor is $\begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix} = 2 \cdot 2 - 1 \cdot 1 = 3 > 0$
  - Since both minors are positive, the Hessian matrix is positive definite
- Therefore, $f(x, y)$ is strictly convex on R^2



Optimization of Convex Functions

Northeastern University

- In convex functions **every local minimum is also a global minimum**
  - Thus, any descent-based method is guaranteed to converge to the global minimum
- If the function is convex but not strictly convex, it can have multiple global minima
  - Example: $f(x, y) = |x|$
    - It has multiple global minima along the entire line $x = 0$



- If the function is strictly convex, it has at most one global minimum
  - But it can also have none, like the function e^x

Iterative Optimization Methods

Northeastern University

- Start from an initial guess $\mathbf{x}_0$
- Generate a sequence of iterates $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, … that ideally move closer to the optimum
- Each point $\mathbf{x}_{k+1}$ is computed from $\mathbf{x}_k$ using an **update rule**
  - Typically based on information about the function at $\mathbf{x}_k$, such as the gradient $\nabla f(\mathbf{x}_k)$
- A method satifies the **descent property** if the objective decreases at each iteration:

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

- Desired properties of an optimization method:
  - **Accuracy**: should be able to find a solution that is close to the true (local/global) optimum
  - **Efficiency**: should converge to the optimum using as few function evaluations as possible
  - **Robustness**: should perform reliably even when the function is nonconvex or noisy

Convergence Rates

Northeastern University

- Measures how quickly the function values $f(\mathbf{x}_k)$ converge to the optimum $f(\mathbf{x}^*)$
- A method has convergence rate $O(g(k))$ if:

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq Cg(k), \quad \text{for all sufficiently large } k$$

  - for some constant $C > 0$ and some $g(k) \to 0$
- Common convergence rates
  - Linear convergence: $f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq p(f(\mathbf{x}_k) - f(\mathbf{x}^*)), \quad 0 < p < 1$
    - The error decays exponentially over k (convergence rate is p^k) $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq Cp^k$
  - Sublinear convergence: $f(\mathbf{x}_k) - f(\mathbf{x}^*) = \mathcal{O}\left(\frac{1}{k}\right)$
    - The error shrinks slower than any linear rate
  - Superlinear convergence: $\lim_{k\to\infty} \frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} = 0$
    - The error shrinks faster than any linear rate
  - Quadratic convergence: $f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq p(f(\mathbf{x}_k) - f(\mathbf{x}^*))^2$

Convergence Rates

Northeastern University

- Example: We aim to minimize the convex quadratic function

$$f(x) = x^2$$

  - It has a unique minimizer $x^* = 0$ and its minimum value is $f(x^*) = 0$
- We build a sequence of iterates using the update rule:

$$x_{k+1} = \frac{1}{2} x_k$$

- Does the sequence converge to the minimum from any starting point x_0?
  - Yes, the sequence is globally convergent (it is a geometric series with $|r| < 1$)
- What is the convergence rate?

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} = \frac{(0.5x_k)^2}{x_k^2} = \frac{0.25x_k^2}{x_k^2} = \frac{1}{4}$$

  - Since the ratio is a constant < 1, the function converges **linearly** to 0 with rate 0.25^k

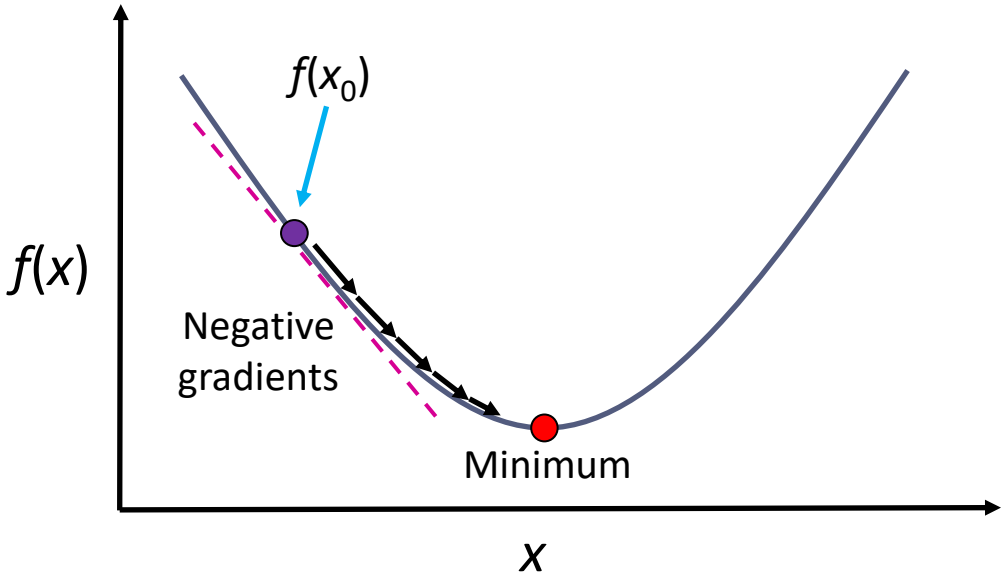
Local vs. Global Convergence

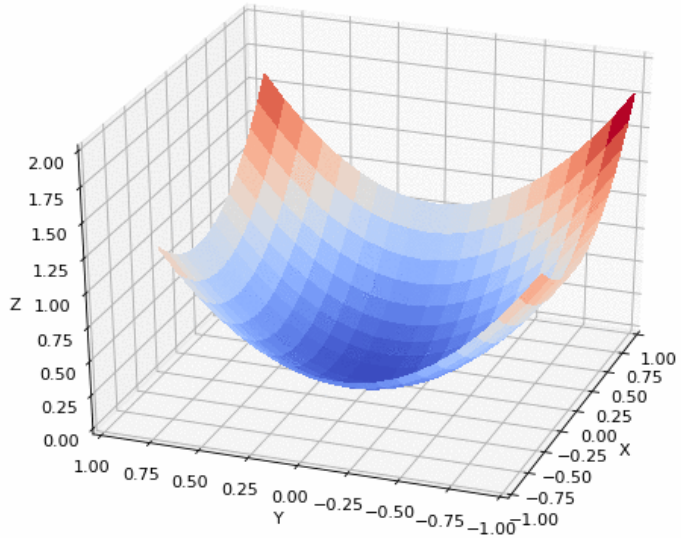
Northeastern University

- **Local convergence**
  - The algorithm converges only when initialized sufficiently close to the optimum
  - No guarantees if starting far from the optimum
  - Often associated with fast convergence rates near the solution (e.g., superlinear rates)
- **Global convergence**
  - The algorithm is guaranteed to converge to a solution from any starting point
    - usually under mild assumptions (e.g., smoothness)
  - Guarantees existence of convergence, but not necessarily rate or speed
- Algorithms may exhibit both local and global convergence properties
  - Some algorithms are globally convergent but become locally faster near the optimum
  - Others may be locally fast but globally unreliable (e.g., Newton's method)

Gradient Descent

- One of the most widely methods for finding a local minimum of a function
- Iteratively moves in the direction opposite to the gradient at the current point $-\nabla f(\mathbf{x})$
  - Moving in this direction leads to the fastest decrease in f (**steepest descent**)
- The update rule is $$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla_{\mathbf{x}} f(\mathbf{x}_k)$$
  - α is a **learning rate** that controls the size of each update ($0 < \alpha \leq 1$)





Gradient Descent Algorithm

Northeastern University

Algorithm A.1 Gradient Descent

Input:

$f(\mathbf{x})$: a differentiable function
$\mathbf{x}_0$: initial value of $\mathbf{x}$
α: learning rate $(0 < \alpha < 1)$
K: maximum number of iterations
ϵ: convergence tolerance

Initialize: $\mathbf{x} \leftarrow \mathbf{x}_0$
for $k = 0$ **to** $K - 1$ **do**
    Compute the gradient at the current point:

$$\mathbf{g}_k \leftarrow \nabla_{\mathbf{x}} f(\mathbf{x}_k)$$

    Update the parameters:

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \mathbf{g}_k$$

    if $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \epsilon$ **then**
        break
return $\mathbf{x}_{k+1}$

Numerical Example

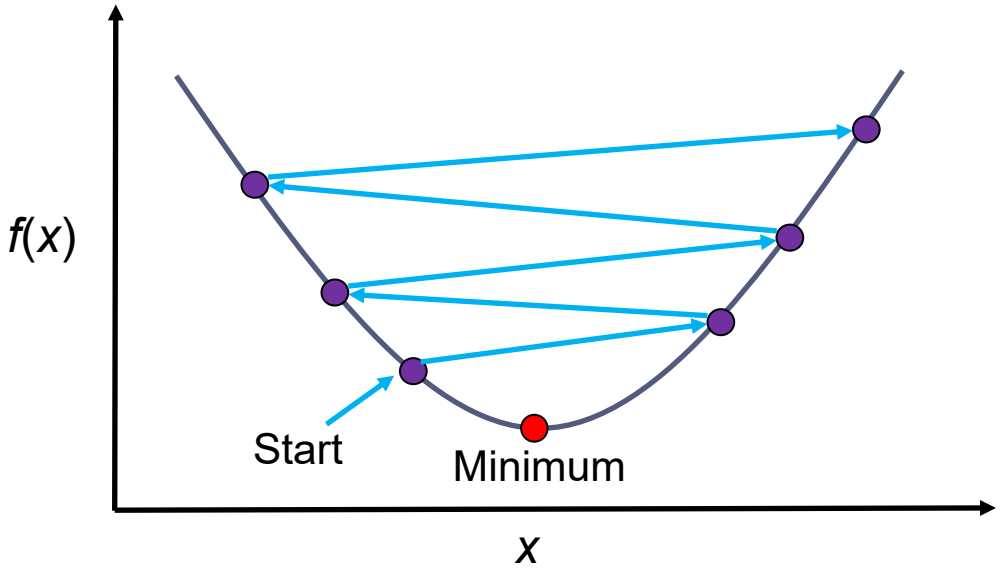
Northeastern University

- Consider the function $f(x) = (x - 2)^2$
  - It has a global minimum at $x = 2$
- The gradient (derivative) of the function is $f'(x) = 2x - 4$
- Applying the gradient descent algorithm starting from $x_0 = 0$ and using $\alpha = 0.3$
- The update rule is: $x_{k+1} = x_k - 0.3 \cdot (2x_k - 4)$
- Running several iterations:

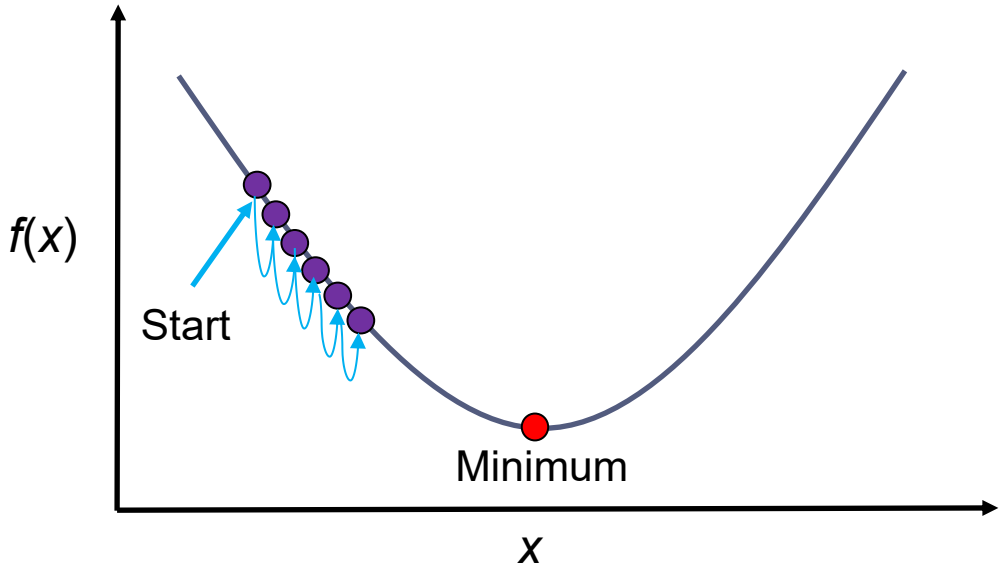
Iteration	x_k	$f'(x_k)$	Update Rule	x_{k+1}
0	0	-4	$0 + 0.3 \cdot 4$	1.2
1	1.2	-1.6	$1.2 + 0.3 \cdot 1.6$	1.68
2	1.68	-0.64	$1.68 + 0.3 \cdot 0.64$	1.872
3	1.872	-0.256	$1.872 + 0.3 \cdot 0.256$	1.9488
4	1.9488	-0.1024	$1.9488 + 0.3 \cdot 0.1024$	1.97952
5	1.97952	-0.04096	$1.97952 + 0.3 \cdot 0.04096$	1.991808
6	1.991808	-0.016384	$1.991808 + 0.3 \cdot 0.016384$	1.9967232

Learning Rate

- The choice of the learning rate α is critical
  - If α is too large, the algorithm might overshoot the minimum, leading to divergence
  - If it is too small, convergence can become very slow



Learning rate too high



Learning rate too small

Line Search

Northeastern University

- Selects the step size that leads to the greatest reduction in the objective function
- Solves a 1D optimization problem along the search direction $\mathbf{d}_k$

$$\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

- Problem: Finding the exact minimizer is typically infeasible or too expensive
- Instead, we can use **inexact line search** methods that guarantee enough reduction in the objective value
  - Backtracking line search
  - Wolfe or Armijo conditions

Learning Rate Schedules

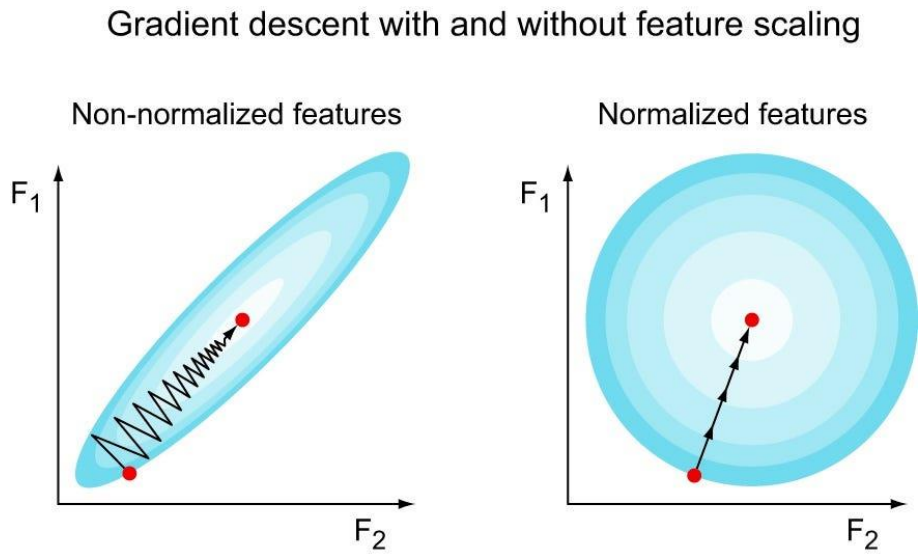
Northeastern University

- Reduce the learning rate over time according to a predefined rule
  - Start with larger steps to speed up progress and escape local minima
  - Use smaller steps in later stages to refine the convergence
- Common learning rate schedules:

Schedule	Description	Formula
Time-based decay	Decreases the learning rate as a function of the iteration number *k*	$\alpha_k = \frac{\alpha_0}{1 + \delta k}$
Step decay	Reduces the learning rate by a constant factor γ every *t* iterations	$\alpha_k = \alpha_0 \cdot \gamma^{\lfloor k/t \rfloor}$
Linear decay	Decreases the learning rate linearly from an initial rate α_0 to a final rate α_τ over τ iterations	$\alpha_k = (1 - \eta)\alpha_0 + \eta\alpha_\tau$
Exponential decay	Reduces the learning rate exponentially over time, where δ controls the decay speed	$\alpha_k = \alpha_0 e^{-\delta k}$

Gradient Descent and Scaling

- Gradient descent can become unstable when input variables are on different scales
  - Takes small steps in some directions and large ones in others
  - Converges very slowly (zig-zagging behavior) or can even diverge
- It is crucial to ensure the input variables (features) are on a similar scale
  - e.g., by using standardization (zero mean, unit variance) or min-max normalization



Convergence Guarantees

Northeastern University

- The convergence behavior of gradient descent depends on properties of the objective function

Assumptions on *f*	Step size	Convergence Type	Rate
Convex, *L*-smooth	$\alpha \leq 1/L$	$f(\mathbf{x}_k) \rightarrow f(\mathbf{x}^*)$	Sublinear: $O(1/k)$
Strongly convex, *L*-smooth	$\alpha \leq 2/L$	$f(\mathbf{x}_k) \rightarrow f(\mathbf{x}^*)$	Linear: $O(p^k)$
Nonconvex + *L*-smooth	Diminishing α_k	$\|\|\nabla f(\mathbf{x}_k)\|\| \rightarrow 0$	Sublinear: $O(1/k)$
Not *L*-smooth		No guarantees	

  - *f* is ***L*-smooth** if its gradient is Lipschitz continuous with constant $L > 0$:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

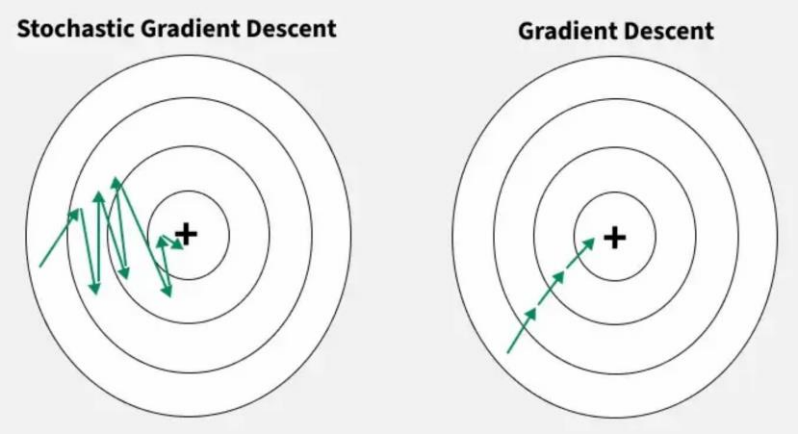
Stochastic Gradient Descent (SGD)

Northeastern University

- An iterative optimization method used to minimize a function when:
  - The objective has a sum structure: $f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$
  - Or only noisy or sampled gradients are available: $f(\mathbf{x}) = \mathbb{E}_{\xi}[f(\mathbf{x};\xi)]$
- Instead of computing $\nabla f(\mathbf{x}_k)$, we use a stochastic approximation $\mathbf{g}_k$ such that

$$\mathbb{E}[\mathbf{g}_k|\mathbf{x}_k] = \nabla f(\mathbf{x}_k)$$

- The update rule is: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k$
- Pros
  - Cheaper per iteration, converges faster to the optimum
  - Scales well to large datasets
- Cons
  - Noisy updates, not guaranteed to converge exactly to the optimum



Empirical Risk Minimization (ERM)

Northeastern University

- In machine learning, we are given a dataset of examples $\{(\mathbf{x}_i, y_i)\}$
- Using a subset of these (**training set**), we build a model $h_{\mathbf{w}}(\mathbf{x})$ with parameters $\mathbf{w}$
- During training, we aim to minimize the average loss of the model over the training set (known as the **empirical risk**)

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, h_{\mathbf{w}}(\mathbf{x}_i))$$

  - n is the number of training examples
  - The **loss function** L measures the discrepancy between the predicted and true labels
- However, our true goal is to minimize the **true risk** (**generalization error**):

$$R(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[L(y, h_{\mathbf{w}}(\mathbf{x}))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y, h_{\mathbf{w}}(\mathbf{x}))p(\mathbf{x}, y)\, d\mathbf{x}\, dy$$

  - $p(\mathbf{x}, \mathbf{y})$ is the (unknown) joint distribution over inputs and labels

Empirical Risk Minimization (ERM)

Northeastern University

- Minimizing $J(\mathbf{w})$ is often challenging:
  - High-dimensional (with thousands to millions of parameters)
  - Nonconvex (with many local minima and saddle points)
  - Expensive to compute exactly (especially for large n)
- To address these challenges, we typically use iterative first-order methods:
  - Batch (full) gradient descent:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}_k)$$

  - SGD uses a randomly sampled example to approximate the gradient

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla_{\mathbf{w}} L(y_i, h_{\mathbf{w_k}}(\mathbf{x}_i))$$

Empirical Risk Minimization (ERM)

Northeastern University

- Training a model using SGD:

Algorithm A.2 Stochastic Gradient Descent (SGD)

Input:

$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$: dataset of training samples
$\mathbf{w}_0$: initial parameter vector
$h_{\mathbf{w}}(\mathbf{x})$: model prediction function
$L(y, \hat{y})$: differentiable loss function
α_k: learning rate schedule

1: $k \leftarrow 0$
2: **while** stopping criterion not met **do**
3: Randomly sample $(\mathbf{x}_i, y_i)$ from D
4: Compute stochastic gradient: $\mathbf{g}_k \leftarrow \nabla_{\mathbf{w}} L(y_i, h_{\mathbf{w}_k}(\mathbf{x}_i))$
5: Update parameters: $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha_k \mathbf{g}_k$
6: $k \leftarrow k + 1$
7: **return** $\mathbf{w}_k$

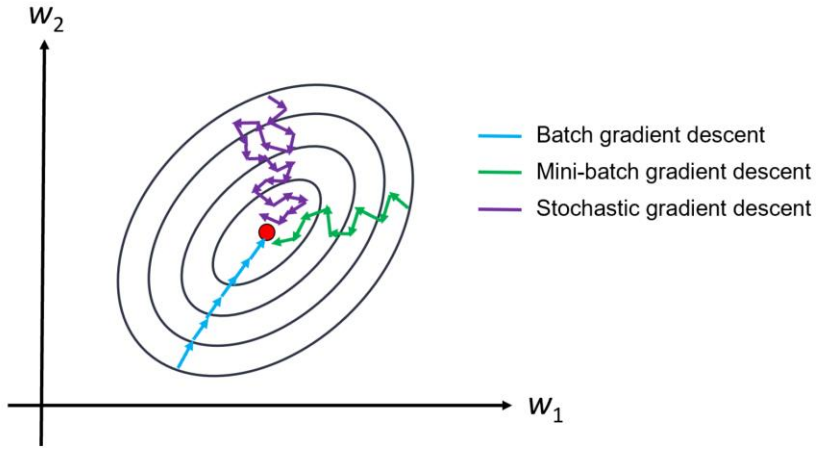
Minibatch Gradient Descent

Northeastern University

- Combines the efficiency of SGD with stability of batch gradient descent
- Gradients are computed on random subsets of the training set called **minibatches**
- Update rule:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\alpha}{|B_k|} \sum_{i \in B_k} \nabla_{\mathbf{w}} L(y_i, h_{\mathbf{w_k}}(\mathbf{x}_i))$$

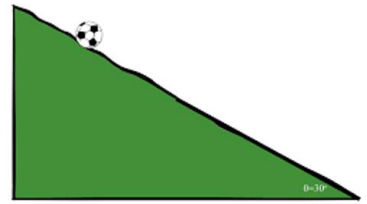
  - B_k is the minibatch size (typically a power of 2 like 32, 64, 128)
- Takes better advantage of parallelized computations (e.g., on GPUs)



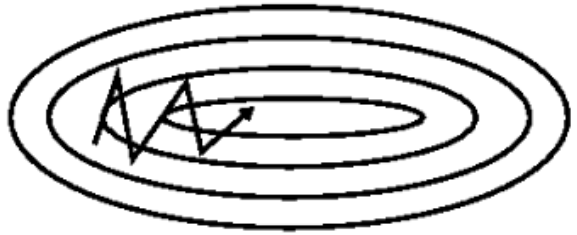
Momentum in Gradient Descent

Northeastern University

- Accumulates gradients of past steps to accelerate movement in consistent directions
  - Acts like a ball rolling down a hill: gains speed in directions with consistent gradient
- Update rules:
  - Velocity update: $\mathbf{v}_{k+1} = \beta \mathbf{v}_k - \alpha \nabla f(\mathbf{x}_k)$
  - Parameter update: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1}$
    - $\beta \in [0, 1)$ is the **momentum coefficient** (e.g., 0.9) and α is the learning rate
- Reduces oscillations and speeds up convergence in narrow valleys
- Helps escape shallow local minima



SGD without momentum



SGD with momentum

Nesterov Accelerated Gradient (NAG)

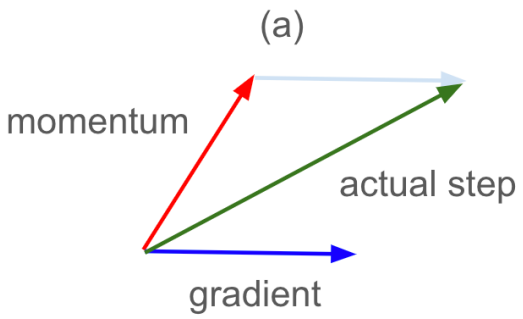
Northeastern University

- Computes the gradients **after** making a step in the direction of the previous velocity
- Update equations:

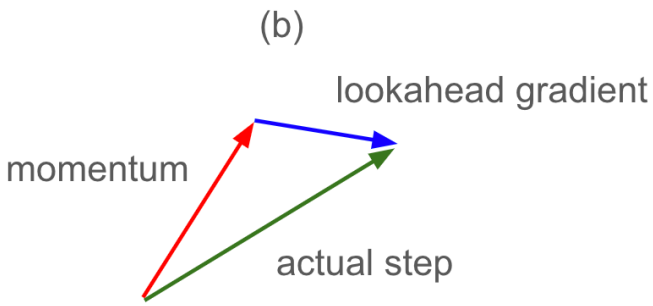
$$\mathbf{v}_{k+1} = \beta \mathbf{v}_k - \alpha \nabla f(\mathbf{x}_k + \beta \mathbf{v}_k)$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1}$$

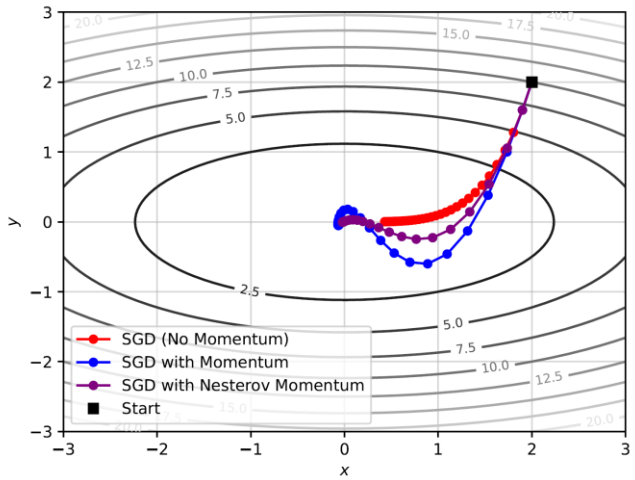
- Improves responsiveness: slows down before overshooting
- Often converges faster than classical momentum
  - For convex functions, achieves an optimal convergence rate of $O(1/k^2)$

Momentum update



Nesterov momentum update





Adaptive Learning Rates

Northeastern University

- Automatically adjust the learning rate during training based on the behavior of the objective function or gradients
- Help improve convergence, especially when
  - The optimal learning rate varies across parameters or iterations
  - The loss surface is ill-conditioned (e.g., narrow valleys, plateaus)
- Key techniques
  - **AdaDelta**: Scales learning rate by inverse square root of accumulated squared gradients
  - **RMSProp**: Uses exponentially decaying average of squared gradients
  - **Adam**: Combines momentum and adaptive scaling of learning rates
- Used mainly in deep learning

Subgradient Descent

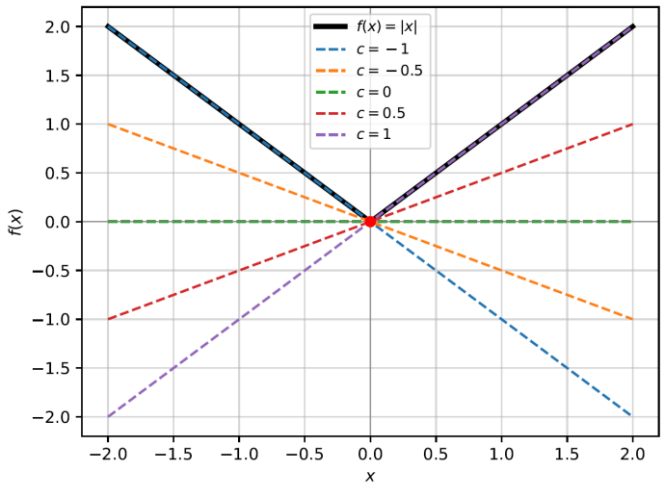
Northeastern University

- Often, the objective function is convex but not differential at some points
- In this case, we can use an extension of the gradient called **subgradient**
- A vector $\mathbf{g} \in R^n$ is called a subgradient of f at point $\mathbf{x}$ if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{y} - \mathbf{x}), \quad \text{for all } \mathbf{y} \in \mathbb{R}^n$$

  - The graph of f lies above or on the line with slope g passing through the point $(x, f(x))$
- The **subdifferential** of at $\mathbf{x}$, denoted by $\partial f(\mathbf{x})$ is the set of all its subgradients at $\mathbf{x}$
- For example, the subdifferential of $f(x) = |x|$ at $x = 0$ is

$$\partial f(0) = [-1, 1]$$



Subgradient Descent

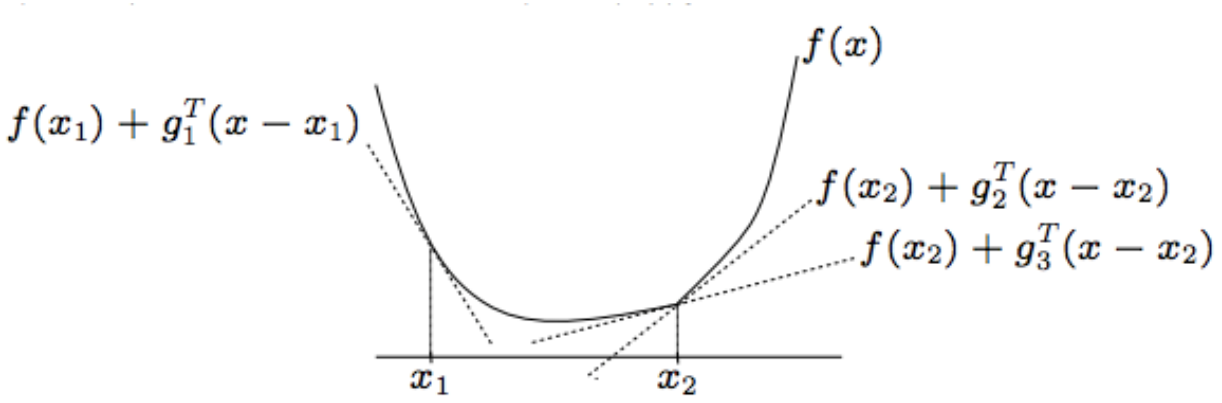
Northeastern University

- The update rule of subgradient descent uses an arbitrary subgradient $\mathbf{g}_k$ at $\mathbf{x}$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k$$

- Unlike gradient descent, this rule may not always yield a descent direction
- Thus, the subgradient method typically tracks the best function value found so far:

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(\mathbf{x}^{(k)})\}$$

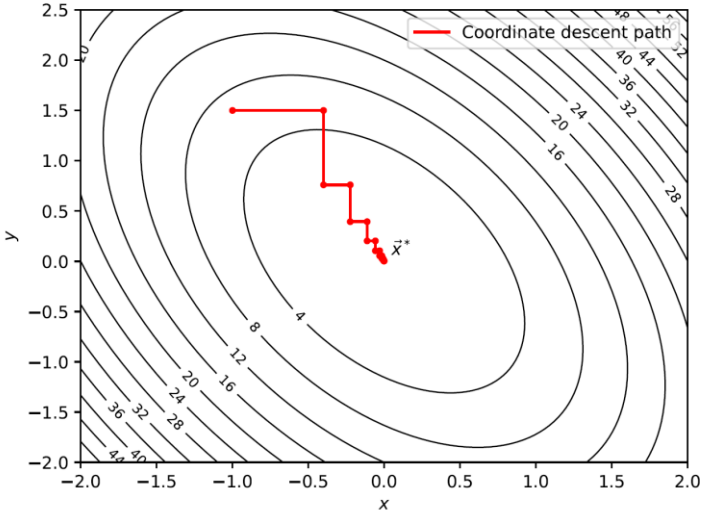


Coordinate Descent

- Minimizes the objective function one coordinate at a time, keeping all others fixed
  - Useful for high-dimensional problems where computing the full gradient is expensive
- In each iteration:
  - Choose one of the coordinates (randomly or in a fixed cyclic order)
  - Solve the one-dimensional optimization problem:

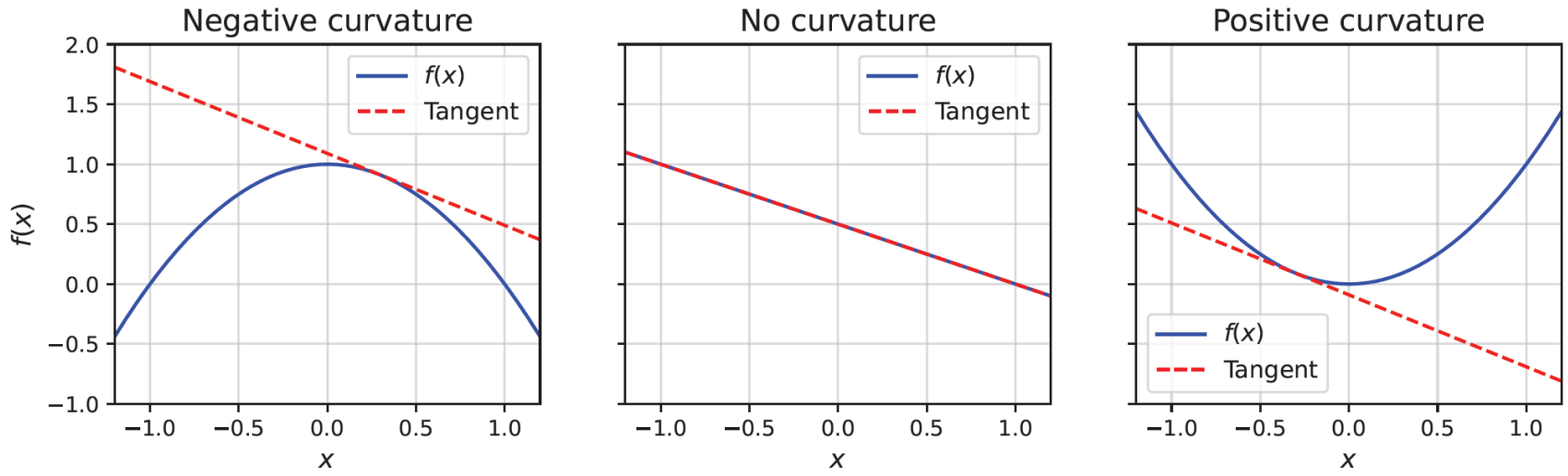
$$x_i^{(k+1)} = \underset{y \in \mathbb{R}}{\operatorname{argmin}} f(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, y, x_{i+1}^{(k)}, \ldots, x_d^{(k)})$$

    - Solved using a closed-form update, line search, or gradient descent
- Under mild conditions, converges to a critical point



Second-Order Methods

- Make use of second-order derivative information to guide the optimization
- Adjust the update direction and step size based on the function's curvature
  - When the function curves gently (gradient changes slowly) ⇒ take larger steps
  - When the function curves sharply (gradient changes rapidly) ⇒ take smaller steps



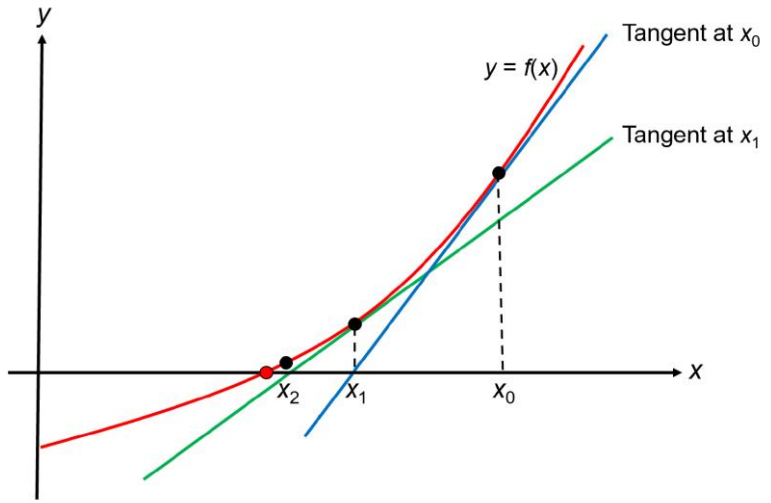
Newton's Method

Northeastern University

- Goal: Find a root of a real-valued function $f(x)$, i.e., solve $f(x) = 0$
- Idea: Use the tangent line to approximate $f(x)$ and iteratively update the guess
- Start from an initial guess x_0 close to the root
- Repeat the update rule:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

  - This corresponds to finding the **x-intercept** of the tangent line at x_k
- Requirements:
  - f must be differentiable
  - $f'(x_k) \neq 0$ during iterations
- Convergence:
  - Quadratic convergence near the root (very fast)
  - May diverge if x_0 is far from the root



Numerical Example

Northeastern University

- Suggest an effective method for finding the square root of a positive number $a > 0$
- We can find the square root by solving the following equation:

$$f(x) = x^2 - a = 0$$

- Using Newton's method, we obtain the update rule:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right)$$

- Example: computing the square root of 10 starting from $x_0 = 3$

Iteration	x_k	$\lvert x_k - \sqrt{10}\rvert$
0	3.00000000	0.16227766
1	3.16666667	0.00438901
2	3.16228070	0.00000304
3	3.16227766	$< 10^{-8}$

  - Quadratic convergence: the number of correct digits nearly doubles in each iteration

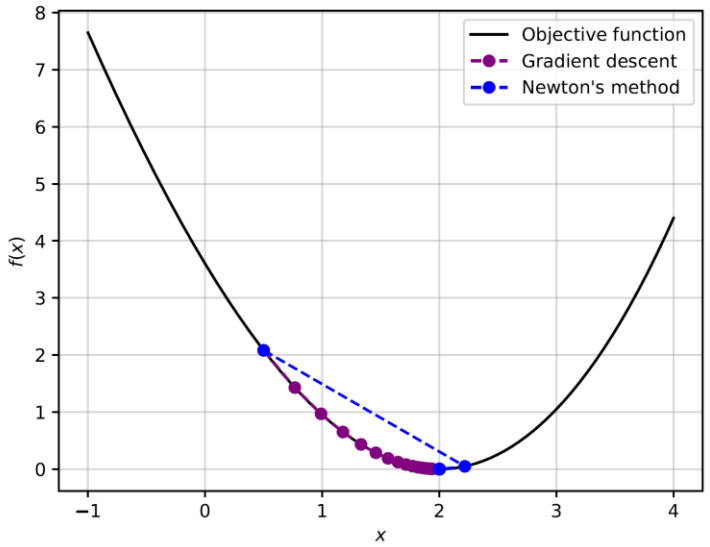
Newton's Method for Optimization

Northeastern University

- At an optimum, the first derivative vanishes: $f'(x^*) = 0$
- This reduces to the problem of finding a root of $f'(x)$
  - Thus, Newton's method can be applied to $f'(x)$ to find such points
- Starting from an initial guess x_0, repeat:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

- Convergence:
  - Converges quadratically near a local minimum if $f''(x^*) > 0$
  - May fail or diverge if $f''(x_k) = 0$



Newton's Method for Optimization

Northeastern University

- Newton's method extends to multivariable functions using the gradient and Hessian:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - H_f(\mathbf{x}_k)^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}_k)$$

- Convergence to a local minimum **x*** requires:
  - f is twice continuously differentiable ($f \in C^2$) near **x***
  - $H_f(\mathbf{x}^*)$ is **positive definite** at **x*** (ensures local minimum)
  - $H_f(\mathbf{x}_k)$ is invertible at each step
  - The initial point $\mathbf{x}_0$ is sufficiently close to **x***
- Under these conditions:
  - Newton's method converges quadratically near **x***
  - Applies to both iterates and function values

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq C \|\mathbf{x}_k - \mathbf{x}^*\|^2$$

Newton's Method for Optimization

Northeastern University

- Pros
  - Fast convergence near the optimum
  - Uses curvature information for better-informed steps (no need for line search)
  - Scale-invariant (no need for feature scaling)
- Cons
  - Computing and inverting the Hessian is expensive in high dimensions
  - May not converge globally without good initialization
  - Can fail if the Hessian is non-invertible or indefinite

Quasi-Newton Methods

Northeastern University

- Approximate the Hessian in Newton's method instead of computing it exactly
  - Maintain fast convergence but with lower computational cost
- Update the Hessian (or its inverse) iteratively using gradient information only
- Use the **secant condition** to ensure the updated matrix captures local curvature:

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k$$

  - B_{k+1} is the updated Hessian approximation
  - $\mathbf{s}_k$ is the step $\quad \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$
  - $\mathbf{y}_k$ is the gradient change $\quad \mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
- For inverse Hessian updates, the secant condition becomes:

$$H_{k+1}\mathbf{y}_k = \mathbf{s}_k$$

- Several algorithms satisfy the secant condition in different ways

BFGS (Broyden-Fletcher-Goldfarb-Shanno)

Northeastern University

- A popular Quasi-Newton method
- Maintains and updates an approximation H_k to the inverse Hessian:

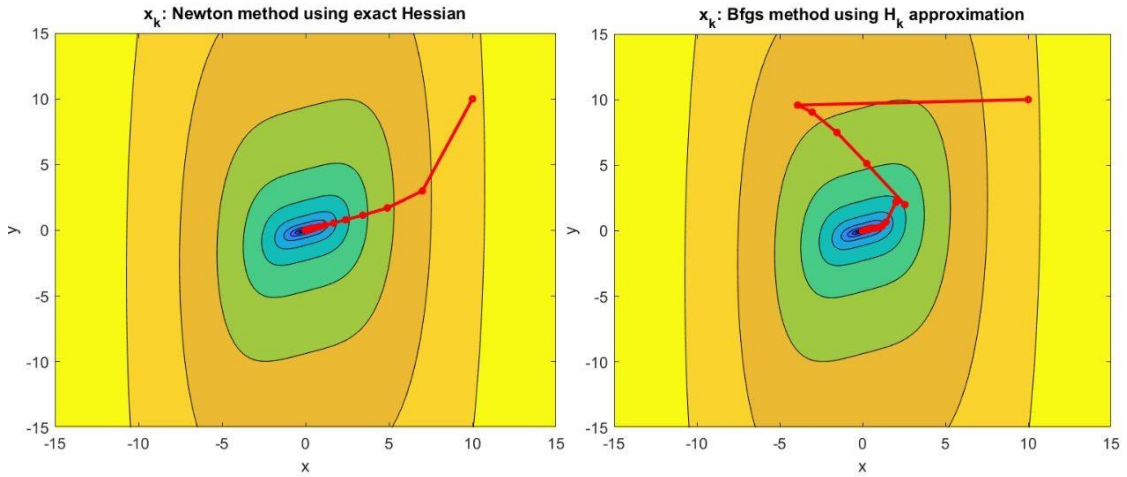
$$H_{k+1} = \left(I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) H_k \left(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$$

  - Preserves the secant condition $H_{k+1}\mathbf{y}_k = \mathbf{s}_k$
  - The difference H_{k+1} - H_k is a sum of two symmetric rank-one matrices (outer products)
    - Requiring only $O(n^2)$ operations to compute the update
  - Minimizes change to H_k: Among all symmetric rank-two updates that satisfy the secant condition, BFGS chooses the one that minimizes $\|H_{k+1} - H_k\|_F$
  - Guarantees that H_{k+1} remains symmetric and positive definite when $\mathbf{y}_k^T \mathbf{s}_k > 0$
    - i.e., when the gradient change has positive curvature

BFGS (Broyden-Fletcher-Goldfarb-Shanno)

Northeastern University

- The rate of convergence is **superlinear**
  - Under the same conditions as Newton's method (e.g, $H(\mathbf{x}^*)$ should be positive definite)
- Example: minimizing the function $f(x, y) = (x - 3y)^2 + x^4$



https://www.linkedin.com/pulse/quasi-newton-methods-bfgs-algorithm-erna-ceka/

  - Newton's method converges in 18 iterations while BFGS converges in 30 iterations

L-BFGS (Limited-memory BFGS)

Northeastern University

- A quasi-Newton method designed for large-scale optimization problems
- Avoids storing the full Hessian approximation:
  - Maintains only the most recent m pairs of update vectors ($\mathbf{s}_k$, $\mathbf{y}_k$)
  - m is typically is within the range 5-20
- Each iteration approximates the inverse Hessian using an efficient two-loop recursion
  - Requires only $O(nm)$ memory and computation per iteration
- Converges superlinearly under standard conditions
  - e.g., smooth objective, positive definite Hessian at local minimizer
- Widely used in machine learning

Constrained Optimization

Northeastern University

- The variables must satisfy a set of equality and/or inequality constraints
- The general form of a constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned}$$

  - $f(\mathbf{x})$ is the objective function
  - $g_i(\mathbf{x})$ are the inequality constraints
  - $h_j(\mathbf{x})$ are the equality constraints
- Key concepts
  - **Feasible set**: The set of points that satisfy all the constraints
  - **Active constraints**: Constraints that are tight (hold as equalities) at the solution

Constrained Optimization

Northeastern University

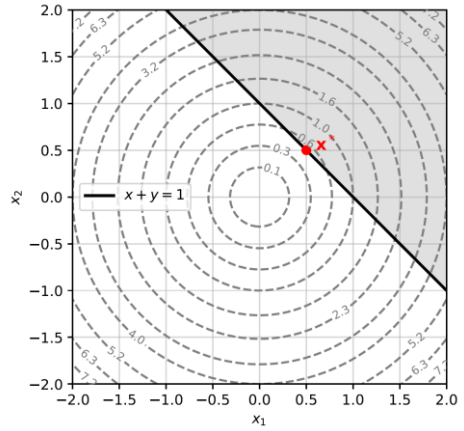
- Our goal is to find a point that minimizes the objective over the feasible region:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{F}}{\operatorname{argmin}} f(\mathbf{x}) \qquad \mathcal{F} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} g_i(\mathbf{x}) \leq 0 & \text{for } i = 1, \ldots, m \\ h_j(\mathbf{x}) = 0 & \text{for } j = 1, \ldots, p \end{array} \right\}$$

  - Note that the gradient $\nabla f(\mathbf{x})$ doesn't have to be 0 at this point!
- For example, what is the optimal solution to the following problem:

$$\begin{aligned} \min_{x,y \in \mathbb{R}} \quad & x^2 + y^2 \\ \text{subject to} \quad & x + y \geq 1 \end{aligned}$$

  - The point on the line $x + y = 1$ that is closest to the origin
  - This is the point (0.5, 0.5)



Example: Diet Optimization

Northeastern University

- A nutritionist is designing a cost-effective daily meal plan using *n* different food items
- Variables and parameters:
  - x_i: Amount of food *i* to include in the diet (e.g., in grams)
  - c_i: Cost per unit of food *i* (e.g., per 100g)
  - a_{ij}: Amount of nutrient *j* in one unit of food *i* (e.g., proteins, vitamins, calories)
  - b_j: Minimum daily requirement for nutrient *j*
  - l_i, u_i: minimum and maximum allowable amount of food *i*
- The optimization problem is:

$$\min_{x_1,\ldots,x_n} \quad \sum_{i=1}^{n} c_i x_i \quad \text{(total cost)}$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_{ij} x_i \geq b_j, \quad j = 1,\ldots,m \quad \text{(nutritional requirements)}$$

$$l_i \leq x_i \leq u_i, \quad i = 1,\ldots,n \quad \text{(portion limits)}$$

Directional Derivatives and Level Sets

Northeastern University

- **Directional derivative** measures how fast f changes in the direction of a unit vector $\mathbf{v}$

$$D_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x})^T \mathbf{v} = \|\nabla f(\mathbf{x})\| \cos\theta$$

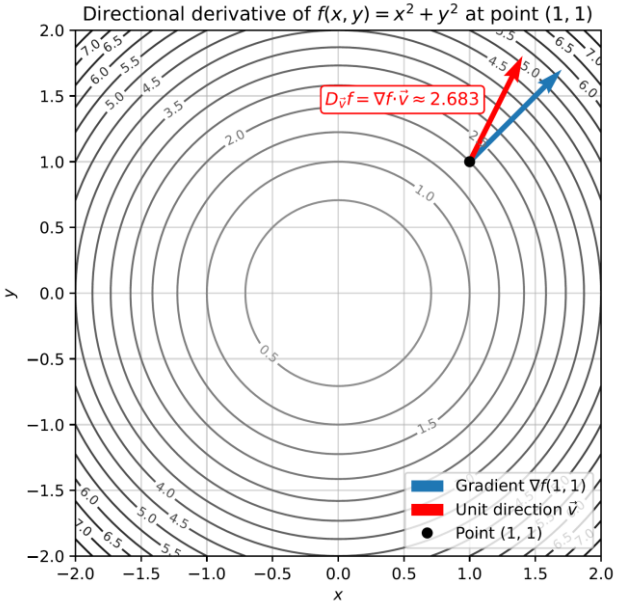
- A **level set** is defined by $f(\mathbf{x}) = c$, where f remains constant
- Let $\mathbf{v}$ be a direction tangent to the level set at $\mathbf{x}_0$
  - Then $f(\mathbf{x}_0 + t\mathbf{v}) = c$ for small t
- Consider the scalar function $g(t) = f(\mathbf{x}_0 + t\mathbf{v})$
  - $g(t) = c$ for all t in some neighborhood of 0, thus $g'(0) = 0$
  - Using the chain rule:

$$g'(t) = \frac{d}{dt} f(\mathbf{x}_0 + t\mathbf{v}) = \nabla f(\mathbf{x}_0 + t\mathbf{v})^T \mathbf{v}$$

  - At $t = 0$:

$$\frac{d}{dt} f(\mathbf{x}_0 + t\mathbf{v})\Big|_{t=0} = \nabla f(\mathbf{x}_0)^T \mathbf{v} = 0$$

- Conclusion: The gradient is **normal** to the level set $f(\mathbf{x}) = c$



The Method of Lagrange Multipliers

Northeastern University

- Method for solving optimization problems subject to equality constraints

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned}$$

- Key idea: Convert the constrained problem into an unconstrained optimization:
  - Introduce a **Lagrange multiplier** λ_j for each equality constraint
  - Define the **Lagrangian function**

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^{p} \lambda_j h_j(\mathbf{x})$$

    - $\boldsymbol{\lambda} \in \mathrm{R}^p$ is a vector that contains the Lagrange multipliers
    - In maximization problems, the constraints are subtracted instead of added
  - Optimal solutions must be stationary points of the Lagrangian:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{0}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = \mathbf{0}$$

Single Equality Constraint

Northeastern University

- At a local optimum, $\nabla f(\mathbf{x}^*)$ must align with the gradient of the constraint $\nabla h(\mathbf{x}^*)$:
  - Let $\mathbf{x}^*$ be a local minimum that satisfies the constraint $h(\mathbf{x}^*) = 0$
  - Any feasible direction $\mathbf{d}$ that stays on the constraint surface must lie in the tangent space:

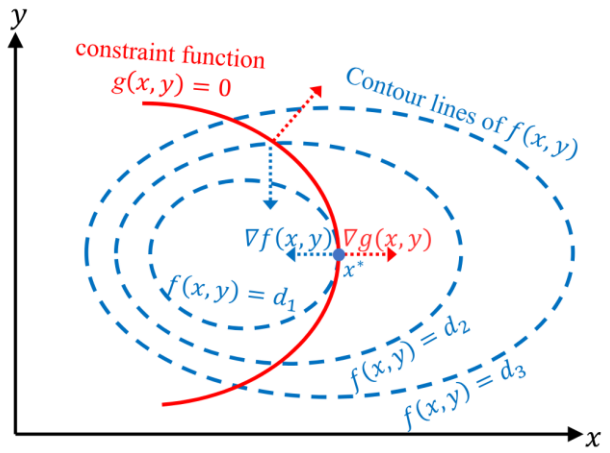
$$\nabla h(\mathbf{x}^*)^T \mathbf{d} = 0$$

  - A direction that would decrease the objective must satisfy: $\nabla f(\mathbf{x}^*)^T \mathbf{d} < 0$
  - At the optimum, no feasible direction can reduce f, so

$$\nabla f(\mathbf{x}^*)^T \mathbf{d} = 0, \quad \text{for all feasible directions } \mathbf{d}$$

  - Thus, $\nabla f(\mathbf{x}^*)$ must be orthogonal to the tangent space
    - Hence, it must lie in the normal space, spanned by $\nabla h(\mathbf{x}^*)$
  - Therefore, there exists λ such that $\nabla f(\mathbf{x}^*) = \lambda \nabla h(\mathbf{x}^*)$
  - This is exactly the stationarity condition from the Lagrangian:

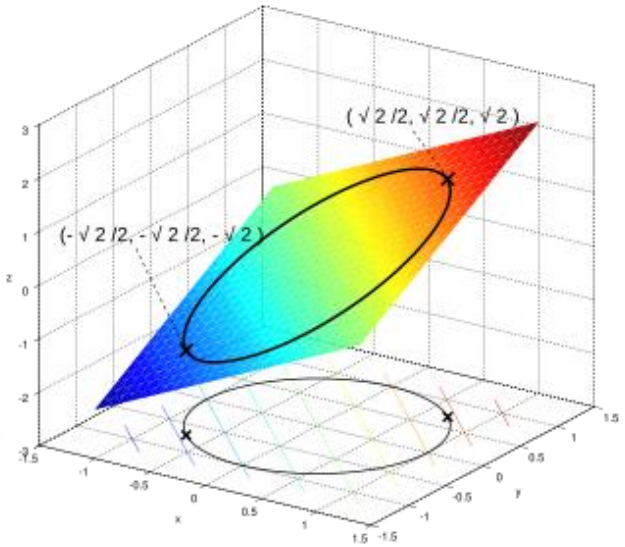
$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) - \lambda \nabla h(\mathbf{x}) = \mathbf{0}$$



Example

- We want to maximize the function $f(x,y) = x + y$
  - subject to the constraint $h(x,y) = x^2 + y^2 - 1 = 0$
- We define the Lagrangian: $\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda h(x,y) = x + y - \lambda(x^2 + y^2 - 1)$
- The gradient of *L* is: $\nabla_{x,y,\lambda}\mathcal{L}(x,y,\lambda) = \left(\frac{\partial \mathcal{L}}{\partial x}, \frac{\partial \mathcal{L}}{\partial y}, \frac{\partial \mathcal{L}}{\partial \lambda}\right) = (1 - 2\lambda x, 1 - 2\lambda y, x^2 + y^2 - 1)$
- Setting the gradient to zero gives the system: $\begin{cases} 1 - 2\lambda x = 0 \\ 1 - 2\lambda y = 0 \\ x^2 + y^2 - 1 = 0 \end{cases}$
- Thus, the two critical points are: $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right), \quad \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$
- The function values at these points are: $f\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) = \sqrt{2} \quad \text{(maximum)}$

$$f\left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right) = -\sqrt{2} \quad \text{(minimum)}$$



Multiple Equality Constraints

Northeastern University

- Now consider multiple equality constraints $h_1(\mathbf{x}) = 0, \ldots, h_p(\mathbf{x}) = 0$
- At a local optimum, the gradient of *f* must be a linear combination of the constraint gradients:

$$\nabla f(\mathbf{x}) = \sum_{j=1}^{p} \lambda_j \nabla h_j(\mathbf{x})$$

  - The constraint equations define an (*n* – *m*)-dimensional surface embedded in R^n
  - The gradients of the constraints are normal to this surface at **x***
  - The tangent space at **x*** consists of all directions **v** ∈ R^n that satisfy $\nabla h_j(\mathbf{x}^*)^T \mathbf{v} = 0$
  - At a local optimum, any feasible direction **v** should not decrease the value of *f*, so:

$$\nabla f(\mathbf{x}^*)^T \mathbf{v} = 0, \quad \text{for all feasible directions } \mathbf{v}$$

  - Therefore, ∇f must be orthogonal to the tangent spaceit lies in the normal space, which is spanned by the constraint gradients $\nabla f(\mathbf{x}^*) \in \text{span}\{\nabla h_1(\mathbf{x}^*), \ldots, \nabla h_p(\mathbf{x}^*)\}$
  - This is the stationarity condition from the Lagrangian: $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) - \sum_{j=1}^{p} \lambda_j \nabla h_j(\mathbf{x}) = \mathbf{0}$

Example

Northeastern University

- Solve the following optimization problem:

$$\begin{aligned} \max_{x,y,z} \quad & f(x,y,z) = xyz \\ \text{subject to} \quad & x+y+z = 32 \\ & x-y+z = 0 \end{aligned}$$

- Define the Lagrangian: $\mathcal{L}(x,y,z,\lambda,\mu) = xyz - \lambda(x+y+z-32) - \mu(x-y+z)$
- Compute the partial derivatives and setting them to zero:

$$\left.\begin{aligned} \frac{\partial \mathcal{L}}{\partial x} &= yz - \lambda - \mu = 0 \\ \frac{\partial \mathcal{L}}{\partial y} &= xz - \lambda + \mu = 0 \\ \frac{\partial \mathcal{L}}{\partial z} &= xy - \lambda - \mu = 0 \end{aligned}\right] \quad yz = \lambda + \mu = xy \;\Rightarrow\; y(z-x) = 0 \;\Rightarrow\; z = x \text{ (assuming } y \neq 0)$$

Example

Northeastern University

- Substitute into the constraints:

$$x - y + z = 0 \;\Rightarrow\; x - y + x = 0 \;\Rightarrow\; 2x = y \;\Rightarrow\; y = 2x$$
$$x + y + z = 32 \;\Rightarrow\; x + 2x + x = 4x = 32 \;\Rightarrow\; x = 8$$

- Then: $y = 2x = 16, \quad z = x = 8$
- The objective value is: $f(x, y, z) = xyz = 8 \cdot 16 \cdot 8 = 1024$
- This is a global maximum since:
  - The two constraints reduce the feasible set to a one-dimensional line segment

$$x - y + z = 0,\ x + y + z = 32 \quad \Rightarrow \quad y = 16,\ z = 16 - x$$

  - Thus, the objective function is a concave quadratic function of the form

$$f(x) = -16x^2 + 256x$$

Inequality Constraints

Northeastern University

- To solve a constrained optimization problem of the form:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned}$$

- We introduce:
  - Lagrange multipliers $\lambda_i \geq 0$ for the inequality constraints
  - Lagrange multipliers $\mu_j \in \mathrm{R}$ for the equality constraints
- Define the generalized Lagrangian:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x})$$

  - Active constraints (where $g_i(\mathbf{x}^*) = 0$) behave like equality constraints
  - Inactive constraints (where $g_i(\mathbf{x}^*) < 0$) must have $\lambda_i^* = 0$

Karush-Kuhn-Tucker (KKT) Conditions

Northeastern University

- Define a set of first-order necessary conditions for the optimum
- If x* is a local optimum and regularity conditions hold, there exist $\lambda_i \geq 0$, μ_j such that
  - **Stationarity**:

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^{p} \mu_j^* \nabla h_j(\mathbf{x}^*) = 0$$

  - **Primal feasibility**:

$$g_i(\mathbf{x}^*) \leq 0, \quad h_j(\mathbf{x}^*) = 0$$

  - **Dual feasibility**:

$$\lambda_i^* \geq 0$$

  - **Complementary slackness**:

$$\lambda_i^* g_i(\mathbf{x}^*) = 0 \quad \text{for all } 1 \leq i \leq m$$

Example

Northeastern University

- Solve the following constrained optimization problem:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) = x^2 + y^2 \\ \text{subject to} \quad & x + 2y = 6 \\ & y \geq 1 \end{aligned}$$

- The Lagrangian is: $\mathcal{L}(x, y, \lambda, \mu) = x^2 + y^2 + \lambda(1 - y) + \mu(x + 2y - 6)$
- The KKT conditions are:
  - Stationarity: $\frac{\partial \mathcal{L}}{\partial x} = 2x + \mu = 0, \quad \frac{\partial \mathcal{L}}{\partial y} = 2y - \lambda + 2\mu = 0$
  - Primal feasibility: $x + 2y = 6, \quad y \geq 1$
  - Dual feasibility: $\lambda \geq 0$
  - Complementary slackness: $\lambda(1 - y) = 0$

Example

Northeastern University

- Case 1: The inequality constraint is **active**

$$y > 1 \;\Rightarrow\; \lambda = 0$$

  - From the stationarity conditions:

$$2x + \mu = 0 \;\Rightarrow\; \mu = -2x$$
$$2y + 2\mu = 0 \;\Rightarrow\; y = -\mu = 2x$$

  - Substitute into the equality constraint:

$$x + 2y = 6 \;\Rightarrow\; x + 4x = 6 \;\Rightarrow\; x = \frac{6}{5},\; y = \frac{12}{5}$$

    - Since $y > 1$, the point is feasible
  - Objective value:

$$f = x^2 + y^2 = \left(\frac{6}{5}\right)^2 + \left(\frac{12}{5}\right)^2 = \frac{36}{5} = 7.2$$

Example

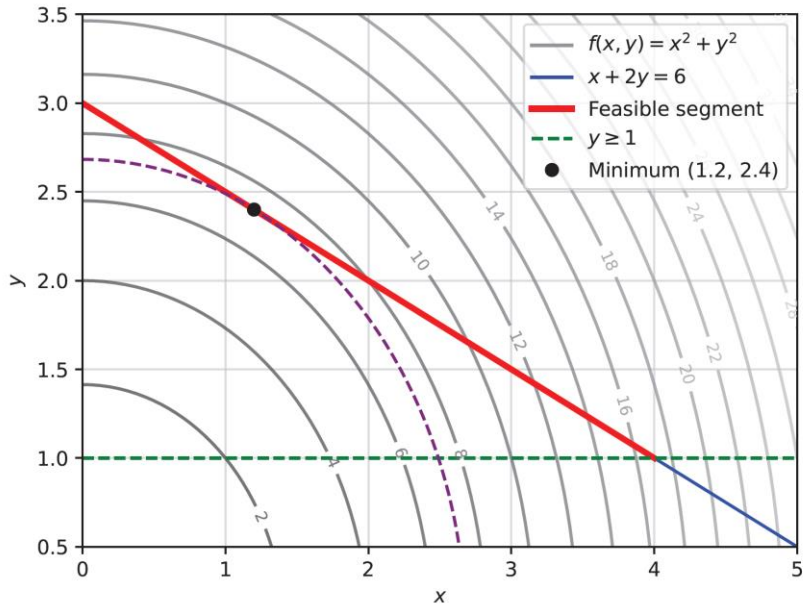
- Case 2: The inequality constraint is **inactive** $y = 1$
  - From the equality constraint: $x + 2 \cdot 1 = 6 \;\Rightarrow\; x = 4$
  - From the stationarity conditions:

$$\mu = -2x = -8$$

$$2y - \lambda + 2\mu = 0 \;\Rightarrow\; 2 \cdot 1 - \lambda + 2 \cdot (-8) = 0 \;\Rightarrow\; \lambda = -14$$

    - But this violates dual feasibility $\lambda \geq 0$
  - Therefore, case 2 is invalid
- Conclusion: the optimal solution is

$$\boxed{x = 1.2,\; y = 2.4,\; f(x, y) = 7.2}$$



Further Readings

- Numerical Optimization by Nocedal and Wright
  - https://www.math.uci.edu/~qnie/Publications/NumericalOptimization.pdf
- Convex Optimization by Boyd and Vandenberghe
  - https://stanford.edu/~boyd/cvxbook/