Large Scale & Distributed Optimization

Angelia Nedić,

January 14, 2022

contact: *Angelia.Nedich@asu.edu; angelianedich@gmail.com*

Lecture 1:

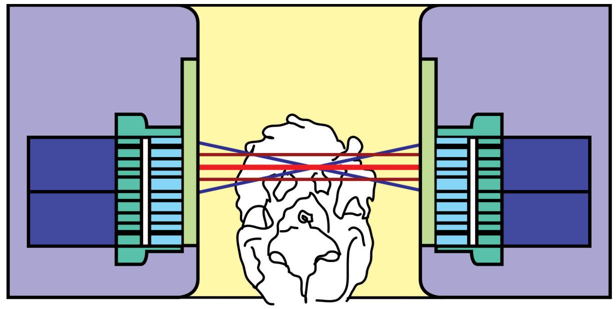
Introduction and Convex Optimization Basics

Large Scale Optimization Problems: Sources

- Automatic Control Systems:
  - Energy Systems
  - Envisioned Smart Grids and Smart Cities
- Signal and Image Processing (Image Reconstruction, Pattern Recognition)
- Bio-science (DNA Sequencing, Protein Sequencing)
- Data Science (Learning from Data)

Image Reconstruction Example

Image Reconstruction in PET-scan [Ben-Tal, 2005]



- Maximum Likelihood Model results in convex optimization

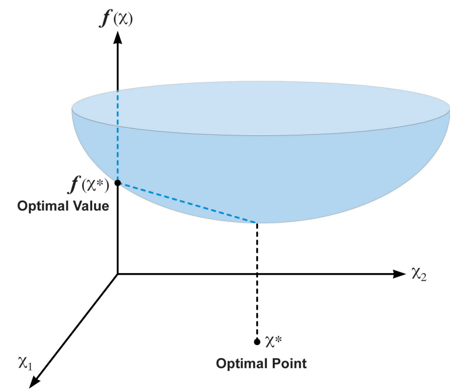
$$\min_{x \geq 0,\ e'x \leq 1} \left\{ -\sum_{i=1}^{m} y_i \ln \left(\sum_{j=1}^{n} p_{ij} x_j \right) \right\}$$

  - $x = (x_1, \ldots, x_n)$ is a decision vector
  - $y = (y_1, \ldots, y_m)$ models measured data (by PET detectors; parameter)
  - p_{ij} probabilities modeling detections of emitted positrons (problem parameters)
  - Depending on the number of pixels in the image, the size of n, m can be in the order of hundreds of thousands.

Least-Squares

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$$

where $\|\cdot\|$ is the Euclidean norm,
$A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.



▶ Using Least-Squares

- In regression analysis, optimal control, parameter estimation

▶ Solving Least-Squares Problems

- Analytical solution: $x^* = (A'A)^{-1}A'b$, where A' denotes the transpose of A
- Reliable and efficient algorithms and software; a mature technology
- Computation time proportional to $n^2 m$ $(A \in \mathbb{R}^{m \times n})$; less if structured

▶ The existing software does not work when n is large!

Machine Learning Problem

- Consider a prototype problem arising in the supervised learning, where a machine (or neural net) is trained from a large data set.
- The problem typically consists of minimizing some objective cost subject to a large number of constraints of the following form:

$$\begin{aligned} \text{minimize} \quad & \rho(x) \\ \text{subject to} \quad & g(x; y_i, z_i) \leq 0, \qquad i = 1, \ldots, m, \quad x \in \mathbb{R}^n, \end{aligned} \tag{1}$$

  where p is the number of data points ($m \gg 1$), $x \in \mathbb{R}^n$ is a decision vector (the vector of weights in neural-nets), and the function $\rho(\cdot)$ is used to promote certain properties of the solutions, such as sparsity or robustness.
- The function $g(x; y_i, z_i)$ represents a constraint imposed by the data point $(y_i, z_i) \in \mathbb{R}^{n+1}$, where y_i is a measurement and z_i is the label associated with the measurement.
- For example, for linear classifiers, each data constraint is linear, i.e.,

$$g(x; y_i, z_i) = 1 - z_i \langle y_i, x \rangle$$

  while the labels z_i are binary.
- The difficulty in solving problem (1) lies in **the large number m of constraints**.

Strategies

- The existing methods developed prior to the emergence of such large problems could not cope with such a large scale.
- To cope with the large number of constraints, there are two main conceptual approaches related to problem (1)
  - **Penalty-Based Reformulation**, which essentially replaces problem (1) with an unconstrained problem obtained by penalizing the constraints to form a new objective function. *The resulting unconstrained problem is not necessarily equivalent to the original constrained problem* (1).
  - **Sampled-Constraint Approximation**, where the problem is addressed directly by sampling the constraints "on-the-go" (within an algorithm).
- We will explore both options in the course.
- Problems with a large number of decision variables, i.e., large n for the decision vector $x \in \mathbb{R}^n$, are not considered. These are typically addressed by block-coordinate approaches, where x is decomposed in blocks of variables (updated in a cyclic or random block-coordinate manner for a given algorithm).

Penalty-Based Reformulation

- Original constrained problem

$$\text{minimize } \rho(x) \qquad \text{subject to} \quad g(x; y_i, z_i) \leq 0, \qquad i = 1, \ldots, m, \quad x \in \mathbb{R}^n,$$

- Introducing a loss function $\ell(\cdot)$ (associated with the quality of data-fitting) and a regularization parameter $r > 0$, the problem is re-formulated as an unconstrained problem:

$$\text{minimize} \quad r\rho(x) + \frac{1}{m}\sum_{i=1}^{m} \ell(x; y_i, z_i), \tag{2}$$

where the loss function penalizes the violation of constraints $g(x; y_i, z_i) \leq 0$, $i = 1, \ldots, m$.

- For example, for linear classifiers, common choices include:
  - The *logistic regression* loss given by $\ell(x; y, z) = \log\left(1 + e^{-z\langle x,y\rangle}\right)$
  - The *hinge loss* $\ell(x; y, z) = \max\{0, 1 - z\langle x, y\rangle\}$.

- By scaling the objective function in (2) with a regularization parameter $r > 0$, we can interpret $1/r$ as the penalty parameter.
- The resulting penalized problem balances the regularizing function $\rho(\cdot)$ and the average sum of the loss functions, where the balance is controlled by the parameter $r > 0$.

Minimizing the Average Sum of Loss-Functions

- We will now consider a general form of the problem in (2):

$$\min_{x \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} f_i(x), \tag{3}$$

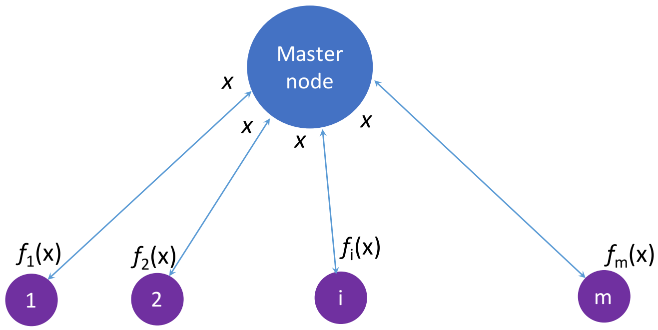
- There is a vast body of work that offers various gradient methods for solving such an unconstrained problems with an additive-type objective function.
- The random incremental gradient method, often referred to as *stochastic gradient descent* in some of the machine learning community, has been the most successful due to its simplicity, and it has a long tradition starting with Kibardin 1980* (see Bertsekas 2012† for an in-depth survey on these methods).
- A renewed interest driven by a desire to improve its convergence rate, which can be unfavorable due to the stochastic errors induced by the sampling of the objective function gradients.
- The development of several efficient variance-reduction methods, such as stochastic variance reduced gradient (SVRG), SAG, SAGA, Katyusha.

*V. M. Kibardin *Decomposition into Functions in the Minimization Problem*, Automation and Remote Control, 40 (9) 1311–1323, 1980

†D. P. Bertsekas *Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey*, in a book on Optimization for Machine Learning, pp. 85–119, MIT Press, Cambridge, MA, 2012

Distributed but Centralized Computational Architecture

- The existing the random incremental gradient methods (aka stochastic gradient descent) can be distributed within a master-slave architecture

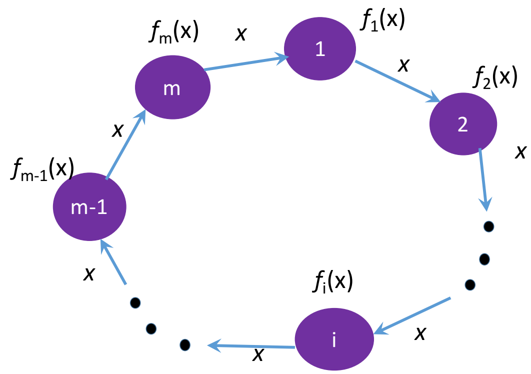


Solving $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^m f_i(x)$ in a master-slave architecture with a master node and m workers. The master node is responsible for maintaining the decision vector x, Each worker i is responsible for processing the function f_i given the state x (typically computes the gradient $\nabla f(x)$).

- Such an architecture is not **fully distributed** (i.e., decentralized) as **it requires a central entity to coordinate the computations** of the slaves (workers).
- This architecture inherently requires the knowledge of the number m of workers.
- Communication with the central entity (master node) is intense when m is large.
- Fast methods (SVRG, SAG, SAGA, etc.) also require master-node with memory of the size $m \times n$ to store past gradients for each $f_i, i = 1, \ldots, m$

Distributed & Decentralized Computational Architecture

- The information processing (iterate updates) of a cyclic incremental gradient method can be interpreted as computations in a cyclic directed graph



Solving $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^m f_i(x)$ with a cyclic incremental method. Each iteration consists of an update of x along a cyclic directed graph over the nodes $1, 2, \ldots, m$. A node i receives x from its up-stream neighbor $i-1$, updates x based on $\nabla f_i(x)$, and sends the updated x to its down-stream neighbor $i+1$.

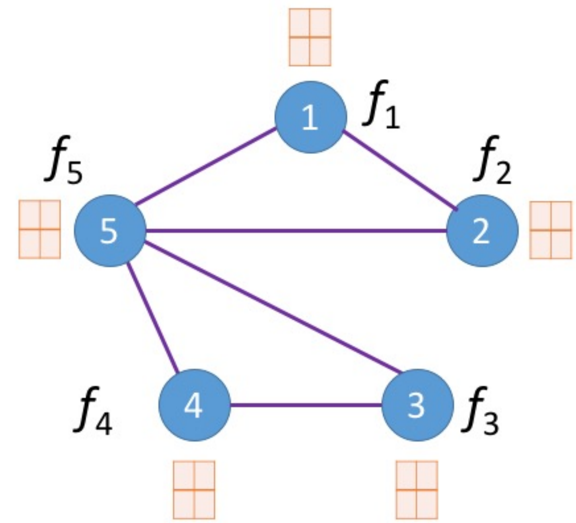
- Information processing (algorithm) along such a cycle has two shortcomings:
  - Takes long time for a full iteration update when m is large
  - Failure of one node, or a link, breaks the computations.

General Distributed & Decentralized Model

We consider a (machine learning) problem

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x)$$

in a system consisting of m agents that are embedded in a communication network.



- Function f_i is *privately known* only to agent i.
- Agents do not share their functions $f_i(\cdot)$'s.
- Agents communicate some limited information with their immediate neighbors only
- The problem is to be solved *distributedly* i.e., **without a central entity**
- Every agent i has only local knowledge of the graph, i.e., it only knows its neighbors
- No agent knows even the total number m of the agents in the system
- **Note:** The problems $\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x)$ and $\min_{x\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(x)$ are equivalent in the sense that their sets of optimal solutions coincide

- The lack of central authority is compensated by agent collaboration and communication
  - DeGroot consensus model [DeGroot 1974] - also referred to as agreement model
  - A variant of this problem, using consensus model, has been studied in the 80's: Borkar & Varaya 1982, Tsitsiklis 1984, Tsitsiklis, Bertsekas & Athens 1986, Bertsekas & Tsitsiklis book "Parallel and Distributed Computations: Numerical Methods" 1989
- We will develop distributed methods by employing gradient methods and DeGroot and Push-sum consensus protocols. To do so we will utilize:
  - Basic graph concepts
  - Row-stochastic and column stochastic matrices (properties of averaging, convergence)
- To address the convex problems we review some basics of Convex Optimization Theory:
  - Basic properties of convex sets and functions
  - Optimality principle

Optimization Theory: Minkowski Sum, Scaling of Sets

- Given a set $X \subset \mathbb{R}^n$ and a scalar $t \in \mathbb{R}$, the scaled set tX is defined by

$$tX = \{tx \mid x \in X\}.$$

- Example: when $t = 0$ and X is not empty, $tX =$? What if X is empty?

- Given two sets $X, Y \subset \mathbb{R}^n$, the (Minkowski) sum set $X + Y$ is defined by

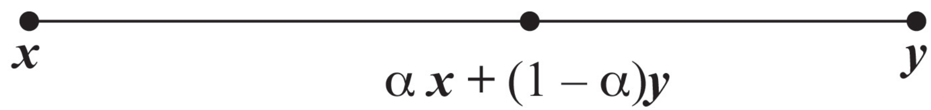
$$X + Y = \{x + y \mid x \in X,\ y \in Y\}.$$

- Example: $X = \{x \in \mathbb{R}^2 \mid x_1 \in \mathbb{R},\ x_2 = 0\}$ and $Y = \{x \in \mathbb{R}^2 \mid x_1 = 0,\ x_2 \in \mathbb{R}\}$. Then, $X + Y$ coincides with the whole space, i.e., $X + Y = \mathbb{R}^2$

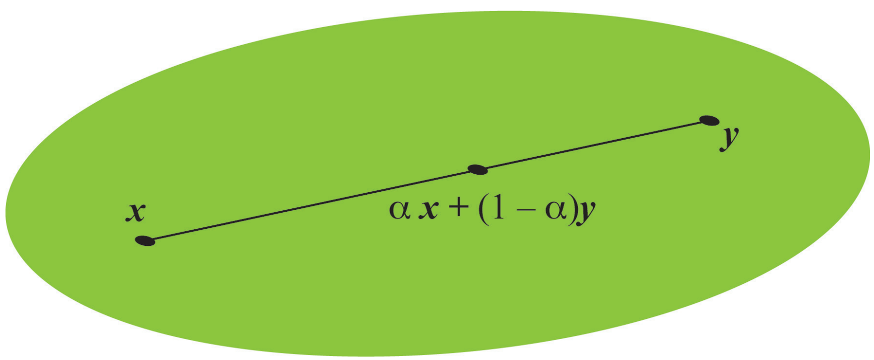
- Question:
  Let $C_1 = \{x \in \mathbb{R}^2 \mid x_1 = 0,\ x_2 \in \mathbb{R}\}$, $C_2 = \{x \in \mathbb{R}^2 \mid x_1 x_2 \geq 1,\ x_1 \geq 0\}$.
  What is $C_1 + C_2$?

Convex Set

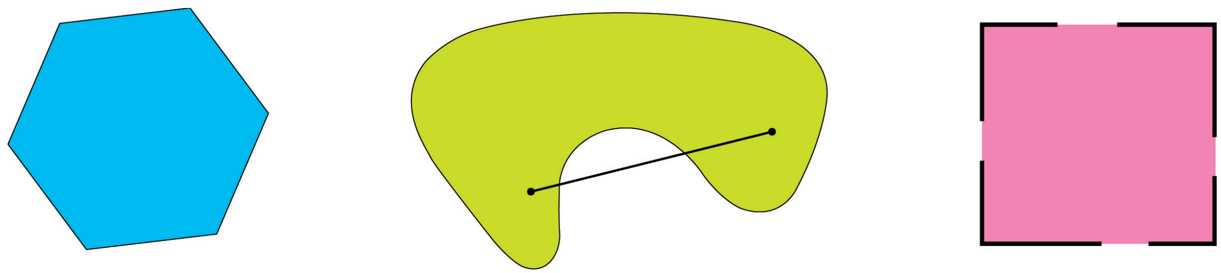
- A *line segment* defined by vectors x and y is the set of points of the form $\alpha x + (1-\alpha)y$ for $\alpha \in [0,1]$



- A set $C \subset \mathbb{R}^n$ is *convex* when, with any two vectors x and y that belong to the set C, the line segment connecting x and y also belongs to C



Examples



Which of the following sets are convex?

- The space $\mathbb{R}^n$
- A *line* through two given vectors x and y
$$l(x,y) = \{z \mid z = x + t(y-x), \quad t \in \mathbb{R}\}$$
- A *ray* defined by a vector x
$$\{z \mid z = \lambda x, \ \lambda \geq 0\}$$
- The *positive orthant* $\{x \in \mathbb{R}^n \mid x \succeq 0\}$ ($\succeq$ componentwise inequality)
- The set $\{x \in \mathbb{R}^2 \mid x_1 > 0, \ x_2 \geq 0\}$
- The set $\{x \in \mathbb{R}^2 \mid x_1 x_2 = 0\}$

Affine Set

- **Def.** A set $C \subset \mathbb{R}^n$ is a *affine* when, with every two distinct vectors $x, y \in C$, the line $\{x + t(y - x) \mid t \in \mathbb{R}\}$ belongs to the set C
  - An affine set is always convex
  - A *subspace* is an affine set

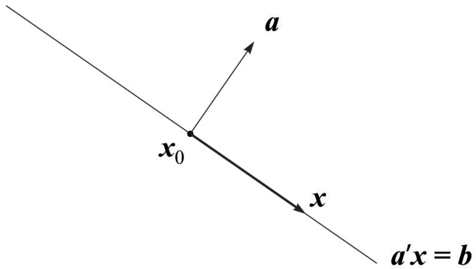
- A set C is affine if and only if C is a translated subspace, i.e., $C = S + x_0$ for some subspace S and some $x_0 \in C$ (representation does not depend on our choice of x_0)

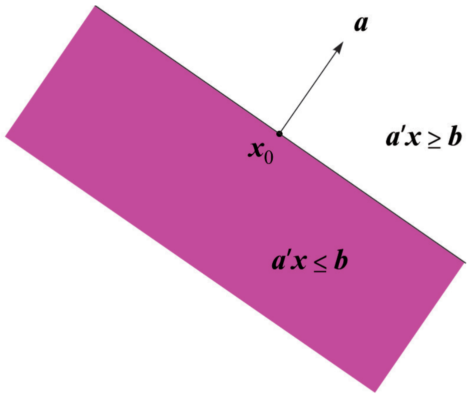
- *Dimension of an affine set* C is the dimension of the subspace S

- **Affine sets are used to define the dimension of a convex set X:** The dimension of a convex set X is the minimal dimension of all affine sets containing the set X.

Hyperplanes and Half-spaces

Hyperplane is a set of the form $\{x \mid \langle a, x\rangle = b\}$ for a nonzero vector a





Half-space is a set of the form $\{x \mid \langle a, x\rangle \leq b\}$ with a nonzero vector a
The vector a is referred to as the *normal vector*

- A hyperplane in $\mathbb{R}^n$ divides the space into two half-spaces

$$\{x \mid \langle a, x\rangle \leq b\} \quad \text{and} \quad \{x \mid \langle a, x\rangle \geq b\}$$

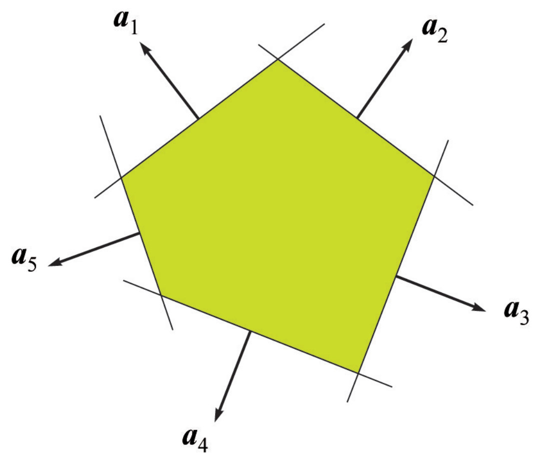
- Half-spaces are convex sets
- Hyperplanes are convex and affine sets

Polyhedral Sets

- A *polyhedral* set is given by finitely many linear inequalities

$$C = \{x \mid Ax \leq b\}$$

where A is an $m \times n$ matrix, $b \in \mathbb{R}^m$, and $Ax \leq b$ is to be understood component-wise.



- Every polyhedral set is convex

Operations Preserving Convexity of Sets

- Given an arbitrary collection of convex sets $\{C_\alpha,\, \alpha \in \ cal A\} \subseteq \mathbb{R}^n$, the *set intersection* $\cap_{\alpha\in\mathcal{A}} C_\alpha$ is convex.
- Given a convex set $C \subseteq \mathbb{R}^n$, the *scaled* set $tC = \{tx \mid x \in C\}$ is convex for any $t \in \mathbb{R}$
- Given two convex sets $C_1 \subseteq \mathbb{R}^n$ and $C_2 \subseteq \mathbb{R}^n$, we have that
  - Their *Minkowski sum* $C_1 + C_2$ is a convex set.
  - Their *Cartesian product* $C_1 \times C_2 = \{(x_1, x_2) \mid x_1 \in C_1,\ x_2 \in C_2\}$ is convex
- Given a convex set $C \subseteq \mathbb{R}^n$ and a partition $x = (x_, x_2) \in \mathbb{R}^{n_1+n_2}$, the *coordinate projection* $\{x_1 \in \mathbb{R}^{n_1} \mid (x_1, x_2) \in C$ for some $x_2\}$ is a convex set
- Given a convex set $C \subseteq \mathbb{R}^n$, its *image* AC under a linear transformation $A : \mathbb{R}^n \to \mathbb{R}^m$ is convex, where
$$AC = \{y \in \mathbb{R}^m \mid y = Ax \text{ for some } x \in C\}$$
- Given a convex set $K \subseteq \mathbb{R}^m$, its *inverse image*[‡] $A^{-1}K$ under a linear transformation $A : \mathbb{R}^n \to \mathbb{R}^m$ is convex, where
$$A^{-1}K = \{x \in \mathbb{R}^n \mid Ax \in K\}$$

[‡] Here, A^{-1} does not denote the inverse of a matrix; its an inverse image of a set K under mapping A

Convex Functions

- Informally: f is convex when for every segment $[x_1, x_2]$, as $x_\alpha = \alpha x_1 + (1-\alpha)x_2$ varies over the line segment $[x_1, x_2]$, the points $(x_\alpha, f(x_\alpha))$ lie below the segment connecting $(x_1, f(x_1))$ and $(x_2, f(x_2))$

- Let f be a function from $\mathbb{R}^n$ to $\mathbb{R}$, $f : \mathbb{R}^n \to \mathbb{R}$

- The domain of f is a set in $\mathbb{R}^n$ defined by
$$\mathrm{dom}(f) = \{x \in \mathbb{R}^n \mid f(x) \text{ is well defined (finite)}\}$$

- **Def.** A function f is *convex* if

(1) Its domain $\mathrm{dom}(f)$ is a convex set in $\mathbb{R}^n$ and

(2) For all $x_1, x_2 \in \mathrm{dom}(f)$ and $\alpha \in [0, 1]$
$$f(\alpha x_1 + (1-\alpha)x_2) \le \alpha f(x_1) + (1-\alpha)f(x_2)$$

Examples: Affine Functions and Norms

- Affine functions are convex

- Norms are convex

- **Examples on** $\mathbb{R}^n$

  - Affine function is of the form: $f(x) = \langle a, x \rangle + b$ with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$
  - Euclidean, l_1, and l_∞ norms
  - General l_p norms

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \quad \text{for } p \geq 1$$

Some Operations Preserving Convexity; Continuity

- Convexity is preserved under:
  - Positive Scaling: $f(\cdot)$ convex and $c > 0$, then $cf(\cdot)$ is convex
  - Sum: $f_1(\cdot)$ and $f_2(\cdot)$ convex, then $(f_1 + f_2)(\cdot)$ is convex
  - Composition with Affine Mapping: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $f : \mathbb{R}^m \to \mathbb{R}$, then $g(x) = f(Ax + b)$ is convex

- If $f : \mathbb{R}^n \to \mathbb{R}$ (means its domain is entire $\mathbb{R}^n$), then $f(\cdot)$ **is continuous on** $\mathbb{R}^n$!

- If $f(\cdot)$ is a convex function, then for any $\gamma \in \mathbb{R}$ the lower-level set $L\gamma(f)$ of f,

$$L\gamma(f) = \{x \in \mathrm{dom}(f) \mid f(x) \leq \gamma\}$$

is convex.

Closed Sets

- A set $X \subset \mathbb{R}^n$ is **closed** if for any sequence $\{x_k\} \subseteq X$ converging to some point $\widehat{x} \in \mathbb{R}^n$, the limit point $\widehat{x}$ belongs to the set X.

- In simple words, any point $\widehat{x}$ that is asymptotically reachable through a sequence of points in X must lie in X.

- If h is a continuous function, then its lower-level set $L_\gamma(h)$ is closed for any $\gamma \in \mathbb{R}$

- **Consequence of Convexity**: When $f : \mathbb{R}^n \to \mathbb{R}$ is convex (its domain is the entire $\mathbb{R}^n$), then its lower-level set $L_\gamma(f)$ is closed and convex set for any $\gamma \in \mathbb{R}$.

- Throughout the rest of course, we will assume that the functions we work with are defined everywhere; unless clearly stated otherwise.

Optimality Principle

- Consider the constrained convex problem of minimizing a convex function $f : \mathbb{R}^n \to \mathbb{R}$ over a closed and convex set $X \subseteq \mathbb{R}^n$.
- **Optimal Solution**: A point $x^* \in X$ is an optimal solution of the problem if

$$f(x^*) \leq f(x) \qquad \text{for all } x \in X$$

- **Optimality Principle**: When $f(\cdot)$ is continuously differentiable, an optimal point can be characterized as follows: $x^* \in X$ is optimal solution for $\min_{x \in X} f(x)$ if and only if

$$\langle \nabla f(x^*), x - x^* \rangle \geq 0 \qquad \text{for all } x \in X$$

  In the absence of convexity, the preceding condition is necessary but not sufficient, i.e., if $x^* \in X$ is optimal solution for $\min_{x \in X} f(x)$, then the condition holds.
- **Optimality Principle for Affine X**: Assume that the set X is a (nonempty) affine set, $\{X = \{x \in \mathbb{R}^n \mid Ax = b\}$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then, the optimality principle is equivalent to the following statement: $x^* \in X$ is optimal solution for $\min_{Ax=b} f(x)$ if and only if there exists a vector $\lambda \in \mathbb{R}^m$ such that

$$\nabla f(x^*) + A'\lambda = 0$$

  i.e., $\nabla f(x^*)$ lies in the range of A'.

Convex Optimization Literature

- S. Boyd and L. Vandenberghe *Convex Optimization*, available free online

- D.P. Bertsekas, A. Nedić, A.E. Ozdaglar, Vandenberghe *Convex Analysis and Optimization*, Athena Scientific, Belmont, MA, 2003

Graphs

- A graph over $m \geq 2$ nodes is denoted by $\mathbb{G} = ([m], \mathcal{E})$, where $[m] = \{1, 2, \ldots, m\}$ and $\mathcal{E} \subseteq [m] \times [m]$ is the set of edges.
- When a graph $\mathbb{G} = ([m], \mathcal{E})$ is undirected (bidirectional), the graph edges are specified by unordered pairs of distinct nodes $\{i, j\} \in \mathcal{E}$.
- When a graph $\mathbb{G} = ([m], \mathcal{E})$ is directed, the graph edges are specified by ordered pair of distinct nodes $(i, j) \in \mathcal{E}$.
- In what follows, graphs will be used to represent **the information flow** among a set of m agents (also referred to as nodes) communicating over a network with following interpretation of the graph edges:
  - An undirected edge (or a link) $\{i, j\}$ indicates that i can receive from and send information to j, and j can receive from and send the information to i;
  - A directed edge (or a link) (i, j) indicates that i can send information to agent j.
- An undirected graph $\mathbb{G}$ is **connected** if there is a path connecting every two distinct nodes in the graph.
- A directed graph $\mathbb{G}$ is **strongly connected** if there is a directed path connecting each node to every other node in the graph.