Numerical Optimization with Python

Lecture 1: Introduction

Lecture 01: Introduction

- Administrivia
- Motivational examples
- Mathematical formulation of an optimization problem
- A practitioner point of view
- Classification of optimization problems
- General course overview
- Background material – overview of some mathematical concepts

Administrivia

- **Email** Yonathan.Mizrahi@post.idc.ac.il
- **Office hours** default is after class (but many more will be set)
- **Course site** https://moodle.idc.ac.il/2022/course/view.php?id=2201705
- 3 hours weekly
  - 2 hours – lecture: theory, methods, algorithms, examples
  - 1 hour – tutorial: Python programming, exercises and more examples

Administrivia

- Prerequisites (will be reviewed when needed)
  - Linear algebra (vectors and matrices and some geometry)
  - Multivariate calculus (gradients, Hessian, chain rule, Taylor’s theorem)
  - Programming **IS** assumed, but Python language will be taught from scratch
  - Probability, statistics? No previous knowledge assumed but some examples will be easier to understand if you have some background (already one example today)

Administrivia

- Grading:
  - Dry exercises (2-3):
    - Mathematical concepts, small proofs, computation
    - Understanding of algorithms and methods taught in class
  - Programming exercises (2-3)
    - Implementing optimization algorithms
    - Test them on (usually very simple) examples

Administrivia

- Grading (cont.):
  - Course project - presented in class during last three sessions of the semester:
    - Paper/book chapter/advanced method
    - Interesting application
    - Choose from a collection I propose, or: you are more than welcome to bring your own suggestions
    - Challenges you in the literature review, reading technical papers and presentation (in addition to the implementation challenge)

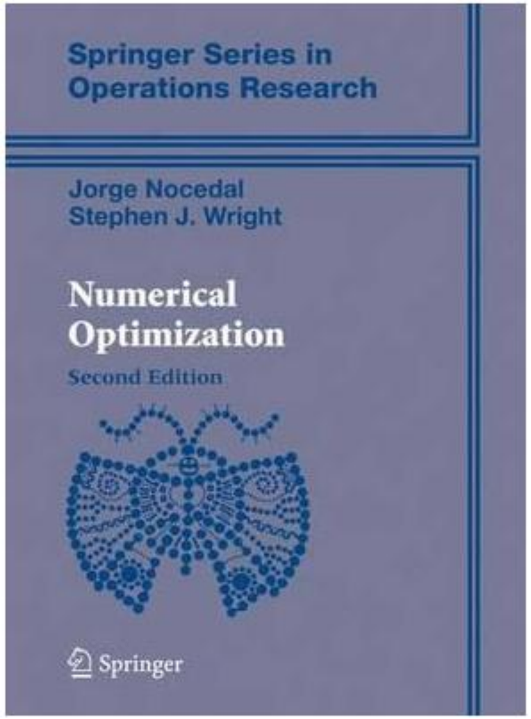
Administrivia

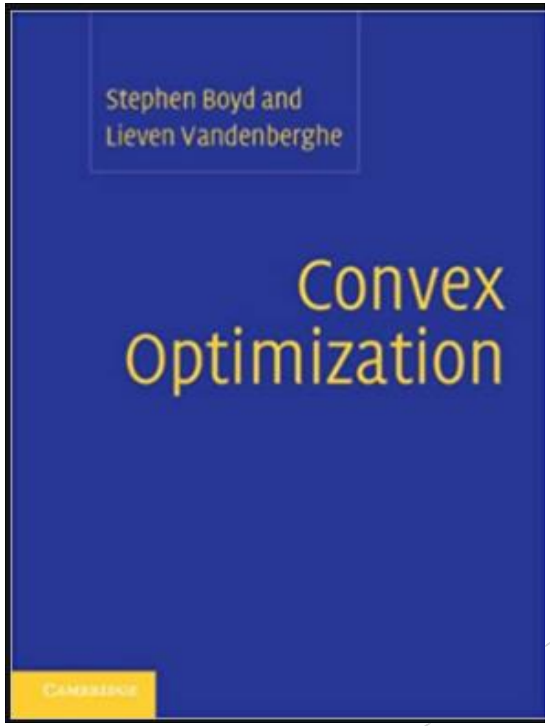
- Grading (cont.):
  - HW assignments – each student submits their own work
  - Working in groups is encouraged, but once team work is done – think and articulate your own solution
  - Project – in pairs

Administrivia

Literature:

- Optimization theory and algorithms will be based on selected chapters from:

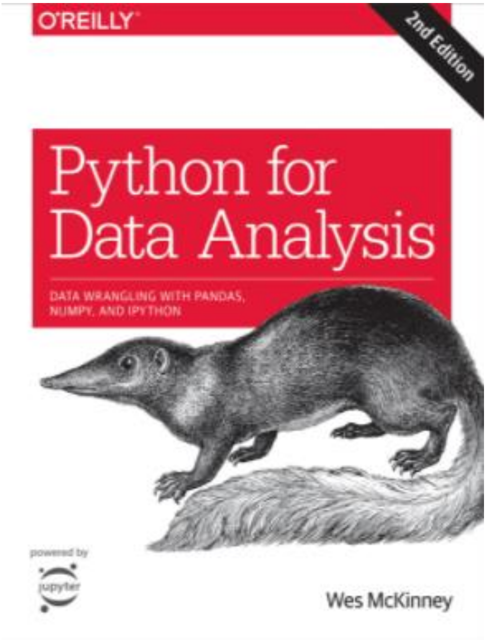




Administrivia

Literature (cont.):

- Python programming topics will be based on selected parts of:



Administrivia

Literature (cont.):

- Other resources we will use from time to time:
  - Python documentation and specific libraries docs/tutorials
  - Other technical papers or book chapters for students selection of project

Motivational Examples

- **Example 0:** what do we already know from basic Calculus?

$$\min_{x\in\mathbb{R}}[e^x + x^2]$$ (unconstrained, single variable)

$$\min_{x\in[a,b]}[e^x + x^2]$$ (closed interval constraints, single variable)

$$\min_{x\in\mathbb{R}^2}\left[e^{x^2+y^2}\right]$$ (unconstrained, multivariate)

$\max[x^2 + 4y^2]$ subject to $x^2 + y^2 = 1$ (eq. constrained, multivariate)

Motivational Examples

- **Example 0:** what do we already know from basic Calculus?

m
$x\in$

So… what's so new in this course??

n
$x\in[a,b]$

$$\min_{x\in\mathbb{R}^2} e^{x^2+y^2} \quad \text{(unconstrained, multivariate)}$$

$$\max x^2 + 4y^2 \ \text{ subject to } x^2 + y^2 = 1 \ \text{(constrained, multivariate)}$$

Motivational Examples

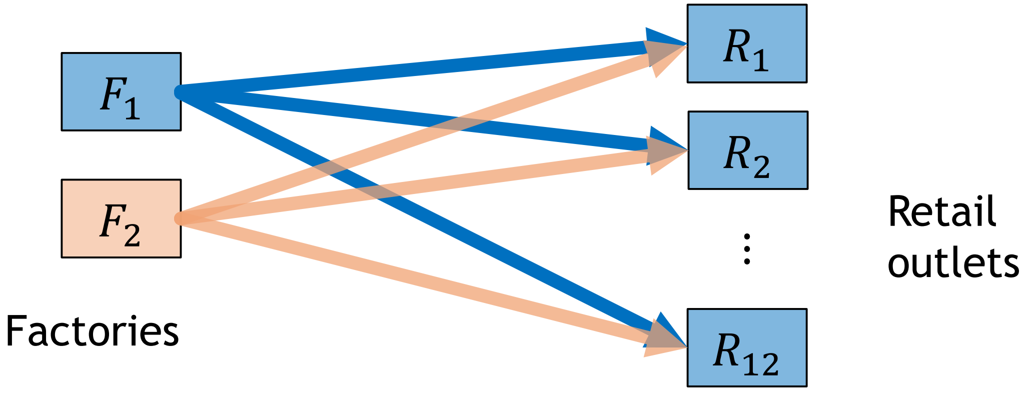
- **Example 0:** what do we already know from basic Calculus?
- What's new:
  - In Calculus courses we could solve ***BY HAND***. This does not happen in real life – we need algorithms and numerical methods
  - We will ***implement*** numerical methods (write code and test it on problems)
  - We will solve families of problems with ***real world applications***
  - We will need more ***advanced theory*** (typically not covered in Calculus courses) to enable a rich family of algorithms

Motivational Examples

- **Example 0:** what do we already know from basic Calculus?
- What's new (cont.):
  - We will account for ***modeling***: how can a story formalized as a mathematical problem that can eventually be solved by an algorithm?
  - We will get some experience with existing optimization software (after we implement some methods on our own)

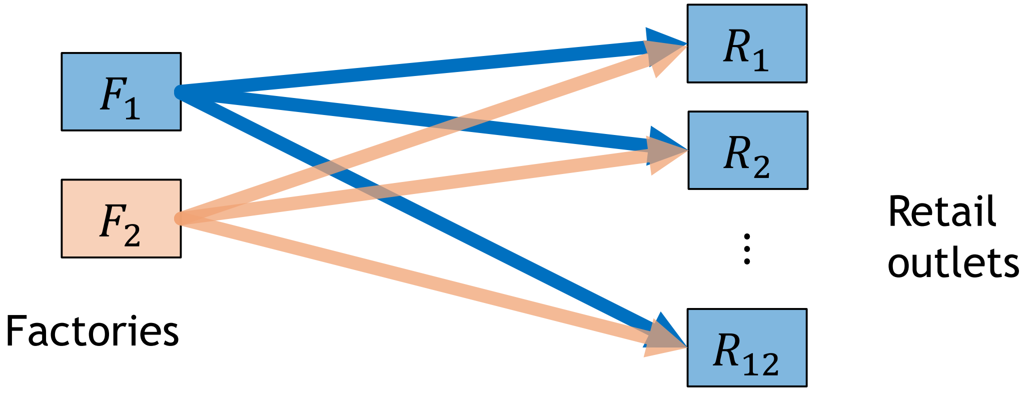
Motivational Examples

- **Example 1:** a transportation problem (adopted from Ch01 of Nocedel & Wright's book)
  - A company has two factories F_1, F_2 and 12 retail outlets $R_1, \dots, R_{12}$
  - Factory i can produce a_i tons per week (the ***capacity*** of the factory)
  - Retail outlet j requires b_j tons per week (the ***demand*** of the retail outlet)



Motivational Examples

- **Example 1:** a transportation problem (adopted from Ch01 of Nocedel & Wright's book)
  - Shipping from factory i to retail outlet j costs c_{ij}
  - **Problem:** *determine how many tons to ship from each factory to each retail outlet such that conditions are satisfied and total cost is minimized*



Motivational Examples

- **Example 1**: a transportation problem (cont.)
  - Denote the unknown variables by x_{ij}: the mount shipped from F_i to R_j
  - The total cost to minimize is then:

$$c_{1,1}x_{1,1} + \cdots + c_{1,12}x_{1,12} + c_{2,1}x_{2,1} + \cdots + c_{2,12}x_{2,12} = \sum_{ij} c_{ij}x_{ij}$$

  - The two capacity constraints are: $\sum_{j=1}^{12} x_{ij} \leq a_i$ for $i = 1,2$
  - The 12 demand constraints are: $\sum_{i=1}^{2} x_{ij} \geq b_j$ for $j = 1,2, \dots, 12$
  - Non-negativity constraints: $x_{ij} \geq 0$ for all i, j

Motivational Examples

- **Example 1:** a transportation problem (cont.)
  - We arrived at the following formulation:

$$\min \sum_{ij} c_{ij} x_{ij}$$

Subject to:

$$\sum_{j=1}^{12} x_{ij} \leq a_i \text{ for } i = 1,2$$

$$\sum_{i=1}^{2} x_{ij} \geq b_j \text{ for } j = 1,2,\ldots,12$$

$$x_{ij} \geq 0 \text{ for all } i,j$$

Motivational Examples

- **Example 1:** a transportation problem (cont.)
  - We arrived at the following formulation:

$$\min \sum_{ij} c_{ij} x_{ij}$$

Subject to:

$$\sum_{j=1}^{12} x_{ij} \leq a_i \text{ for } i = 1,2$$

$$\sum_{i=1}^{2} x_{ij} \geq b_j \text{ for } j = 1,2,\ldots,12$$

$$x_{ij} \geq 0 \text{ for all } i,j$$

The objective function (the one we minimize) and all constraints are linear in the unknown variables

This type of problem is called a Linear Program (LP)

Motivational Examples

- **Example 2**: slightly more complicated modelling - portfolio optimization (Nocedel & Wright Ch16/Boyd Ch04)
  - To increase expected return. an investor must be willing to tolerate greater risks
  - The tradeoff is modeled in ***portfolio theory***
  - Assume n possible investments with return $r_1, \dots, r_n$.

Fig: The Economic Times https://economictimes.indiatimes.com/wealth/invest/why-you-should-not-try-to-time-the-stock-market/articleshow/64230309.cms?from=mdr

Motivational Examples

- **Example 2:** slightly more complicated modelling - portfolio optimization (Nocedel & Wright Ch16/Boyd Ch04)
  - These are not known, and are modelled as ***random variables*** with expected values $\mu_i = \mathrm{E}[r_i]$ and variance $\sigma_i^2 = \mathrm{E}[(r_i - \mu_i)^2]$
  - If you are not familiar with ***expectation*** and ***variance*** yet, think of the expectations as averages over time of past returns, and of variance as a measure of how much the past returns fluctuate from their average (how volatile)

Fig: The Economic Times https://economictimes.indiatimes.com/wealth/invest/why-you-should-not-try-to-time-the-stock-market/articleshow/64230309.cms?from=mdr

Motivational Examples

- Example 2: portfolio optimization (cont.)
  - A portfolio is a mixture of the investments: $R = \sum_{i=1}^{n} x_i r_i$
  - The x_i's are non-negative weights: $x_1, \dots, x_n \geq 0, \sum_{i=1}^{n} x_i = 1$
  - To model how desirable the portfolio is we need the expected return and variance of the random variable R. We will see in an exercise:

$$\mathrm{E}[R] = \sum_{i=1}^{n} x_i \mu_i = x^T \mu \text{ and } \mathrm{Var}[R] = x^T G x,$$

Where G is the $n \times n$ symmetric matrix with $G_{ij} = \mathrm{E}\left[(r_i - \mu_i)(r_j - \mu_j)\right]$ called the ***covariance matrix***

Motivational Examples

- Example 2: portfolio optimization (cont.)
  - As the name suggests, the covariance matrix $G_{ij} = \mathrm{E}\big[(r_i - \mu_i)(r_j - \mu_j)\big]$ measures the tendency of investments to move in the same direction (co-vary)
  - We are interested in a portfolio with high return and small variance
  - The model by ***Markowitz*** suggests the following formulation:

$$\max[x^T\mu - \kappa x^T G x]$$

Subject to: $\sum_{i=1}^{n} x_i = 1,\ x_i \geq 0$

Motivational Examples

- Example 2: portfolio optimization (cont.)
  - ***Markowitz*** portfolio optimization:

$$\max[x^T\mu - \kappa x^T G x]$$

Subject to: $\sum_{i=1}^{n} x_i = 1,\ x_i \geq 0$

  - The constant κ (Greek letter kappa) is a risk tolerance parameter
  - We arrived at a maximization problem with a quadratic objective and linear constraints: Quadratic Programming (QP), for which we will study algorithms

Motivational Examples

- Example 2: portfolio optimization (cont.)
  - ***Markowitz*** portfolio optimization – alternative formulation:

$$\min x^T G x$$

Subject to: $$\begin{cases} x^T \mu \geq r_{min} \\ \sum_{i=1}^{n} x_i = 1 \,,\; x_i \geq 0 \end{cases}$$

- In this modeling we have a hard constraint on minimal return, and minimize (softly) the risk objective (problems are not at all equivalent)
- Still qualifies as QP (but what if we had the risk as a hard constraint?)

Motivational Examples

- In examples 1-2 we did not ***solve***
- We focused on ***modelling***: formalizing a real life problem/business use-case as a mathematical problem
- Modelling is obviously not unique
- We could have a much less simple transportation model (storage and manufacturing cost, non-linear price models, etc.)
- We could have chosen many other models to select a portfolio (other objectives, other constraints such as quadratic risk in the constraints)

A Practitioner's Point of View

Customer and/or
Product manager

Present real life problems and business cases, typically:
- Complex
- Non-formally defined
- Missing information or vague
- Hard!

Value?

Technology (R&D)

Modelling:
- Formal problem definition
- Can we model as an optimization problem?
- One that is solvable? Perhaps even solved?
- Simplifications? Relaxations?

Solving:
- Correctness
- Efficiency
- Robustness
- In house vs. commercial solvers
- Ad-hoc vs. general algorithm

Mathematical Formulation

- Denote by $x \in \mathbb{R}^n$ the vector of variables, also called ***unknowns*** or ***parameters***
- f is the ***objective function***: a (scalar) function of x we want to minimize (or maximize)
- We may also have ***constraint functions***, denoted c_i or f_i: scalar functions of x that define equalities and inequalities that the unknown vector x must satisfy

Mathematical Formulation

- In the above notations, the optimization problems we will study can be presented as:

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

Subject to:

$c_i(x) = 0, i \in \mathcal{E}$ (equality constraints)

$c_i(x) \leq 0, i \in \mathcal{I}$ (inequality constraints)

Mathematical Formulation

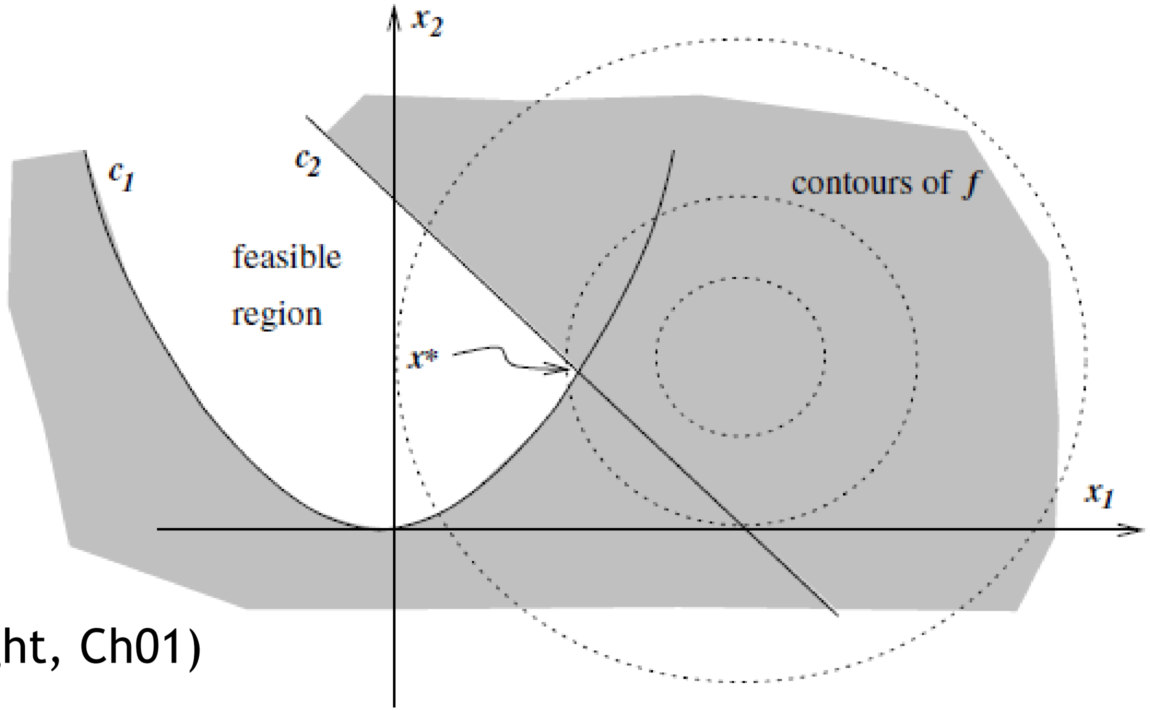
- Example:

$$\min_{x \in \mathbb{R}^2} [(x_1 - 2)^2 + (x_2 - 1)^2]$$

Subject to:

$$x_1^2 - x_2 \leq 0$$

$$x_1 + x_2 \leq 2$$



(Fig: Nocedal & Wright, Ch01)

Mathematical Formulation

- Alternative notations we may use/come across:
  - Max instead of min (can negate the objective to convert)
  - RHS of constraint functions appears as a constant vector b (not necessarily zero)
  - Constraints with $\geq$ instead of $\leq$
- The above formulations are equivalent and easily transform from one to the other by rearranging the equations

Classification of Optimization Problems

- Constrained vs. unconstrained optimization
- Linear vs. nonlinear optimization
- Convex vs. nonconvex optimization
- Local vs. global optimization
- Continuous vs. discrete optimization
- Stochastic vs. deterministic optimization

General Course Overview

- **Week 1:** Introduction, mathematical formulation, some review and basic Python
- **Week 2+3**: Unconstrained optimization – problem definition, necessary and sufficient conditions for local min, convexity (definitions), Line Search algorithms (Gradient Descent, Newton's Method, Quasi-Newton methods
- **Week 4+5:** Constrained Optimization – problem definition, KKT conditions – formulation, Lagrangian function and Lagrange dual problem

General Course Overview

- **Week 6+7:** Lagrange duality, proof of KKT conditions and examples, perturbations and sensitivity analysis
- **Week 8+9:** Algorithms for constrained optimization – Linear KKT, Newton’s method, Interior Point methods (log-barrier method)
- **Week 10**: Beyond convexity: penalty methods and Augmented Lagrangian for local minimization
- **Week 11-13:** Project presentations (~30 min. talk per topic)

Review of Mathematical Background

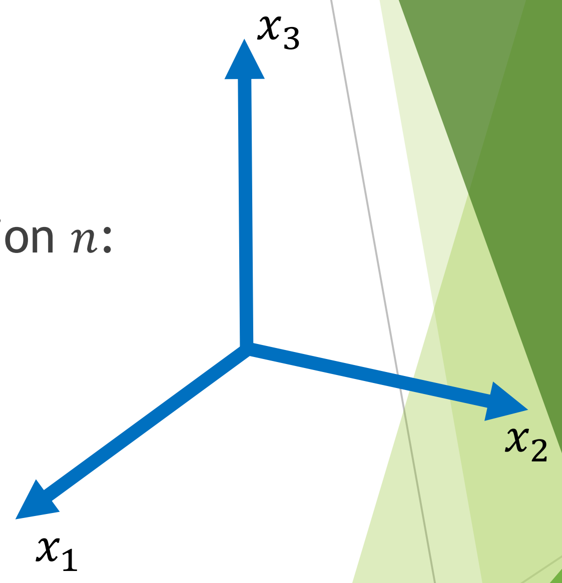
- We now set up mathematical notation for the course, and recall some important facts from Geometry, Linear Algebra and Multivariate Calculus
- Not an exhaustive review
- During the course - other material we will need will be reviewed when used

Review of Mathematical Background

The Euclidean space $\mathbb{R}^n$

- Most of our time will be spent in the real Euclidean space of dimension n:

$$\mathbb{R}^n = \left\{ \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, x_i \in \mathbb{R} \right\}$$



- By convention vectors are columns, unless explicitly stated otherwise (x denotes a column vector, x^T denotes a row vector)
- The ***standard inner product*** on $\mathbb{R}^n$: $\langle x, y \rangle = y^T x = \sum_{i=1}^{n} x_i y_i$

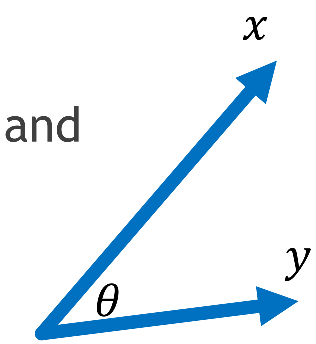
Note: subscript sometimes used for index of a scalar component and sometimes of a point/vector. This will be clear according to context

Review of Mathematical Background

The Euclidean space $\mathbb{R}^n$ (cont.)

- The standard inner product induces the ***Euclidean norm*** (length of a vector):

$$\|x\|_2 := \sqrt{x^T x} = [x_1^2 + \cdots + x_n^2]^{\frac{1}{2}}$$



- (when we omit the 2 and write $\|x\|$ we mean Euclidean norm, and occasionally we will come across other types of norms)
- From Law of Cosines it follows that: $y^T x = \|x\|\|y\| \cos\theta$
- Vectors x, y are called ***orthogonal*** if $x^T y = 0$ $(\theta = 90°)$ and *orthonormal* if $\|x\| = \|y\| = 1$

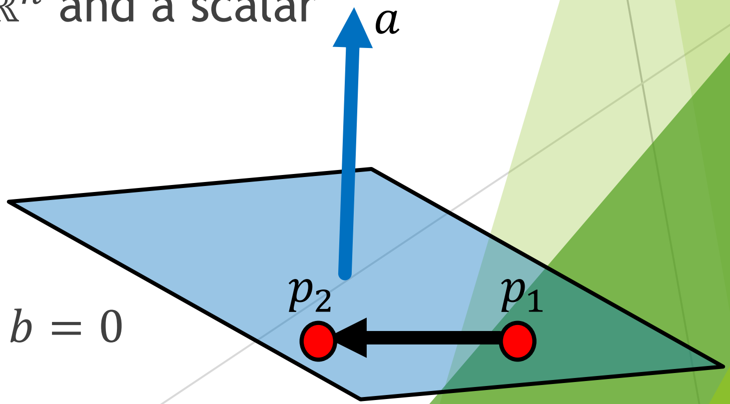
Review of Mathematical Background

The Euclidean space $\mathbb{R}^n$ (cont.)

- An ***affine subspace*** is a translated linear subspace (not necessarily contains the origin)

- A ***hyper-plane*** is an $n-1$ dimensional ***affine subspace*** of $\mathbb{R}^n$ that can be represented by $a^T x = b$ for a constant nonzero vector $a \in \mathbb{R}^n$ and a scalar $b \in \mathbb{R}$

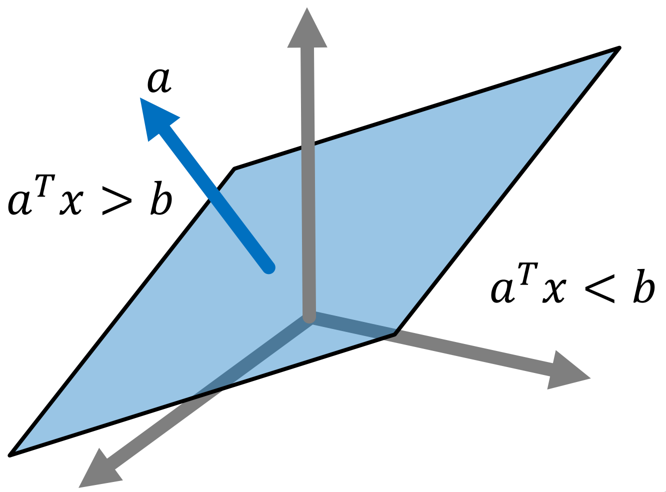
- The vector a is then orthogonal to the hyper-plane, and is called ***the normal*** vector: $a^T(p_2 - p_1) = a^T p_2 - a^T p_1 = b - b = 0$



Review of Mathematical Background

The Euclidean space $\mathbb{R}^n$ (cont.)

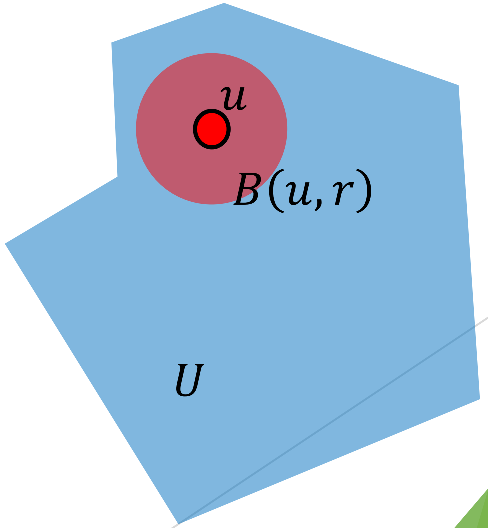
- The hyper-plane $a^T x = b$ separates space into two components: one ***open half-space*** $a^T x > b$, the other ***open half-space*** $a^T x < b$ and $a^T x = b$ is their ***common boundary***
- The normal vector a points in the direction of $a^T x > b$ (why?)



Review of Mathematical Background

<u>**The Euclidean space $\mathbb{R}^n$ (cont.)**</u>

- The ***open ball*** of radius r around x_0: $B(x_0, r) \coloneqq \{x \in \mathbb{R}^n \colon \|x - x_0\| < r\}$
- For the ***closed ball*** the inequality is $\leq$ and denoted $\bar{B}(x_0, r)$
- A subset $U \subset \mathbb{R}^n$ is called ***open*** if for every $u \in U$ there is an open ball around u contained in U
- A subset $V \subset \mathbb{R}^n$ is called ***closed*** if it contains all limit points of all possible sequences of points in V



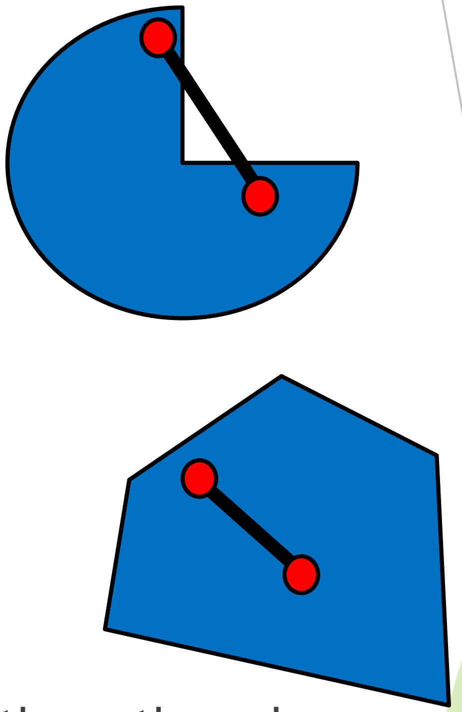
Review of Mathematical Background

The Euclidean space $\mathbb{R}^n$ (cont.)

- A subset $F \subset \mathbb{R}^n$ is ***bounded*** if for all $x \in F$, $\|x\| \leq M$ for some constant $M > 0$
- Some useful facts and definitions:
  - $A \subset \mathbb{R}^n$ is open $\Leftrightarrow$ A^C is closed
  - A subset of $\mathbb{R}^n$ that is closed and bounded is called ***compact***
  - The ***interior*** of A is the largest open set contained in A and is denoted $\text{int}(A)$
  - The ***closure*** of A is the smallest closed set containing A and is denoted $\text{cl}(A)$ (or $\bar{A}$)

Review of Mathematical Background

<u>**The Euclidean space $\mathbb{R}^n$ (cont.)**</u>



- A subset $C \subset \mathbb{R}^n$ is called ***convex*** if for any pair $x, y \in C$, the line segment between x and y is also contained in C
- The line segment can be written as all ***convex combinations*** of x and y: $\alpha x + (1 - \alpha)y$, for all $\alpha \in [0,1]$
- We will study convex sets and ***convex functions*** in further depth, as they play an important role in mathematical optimization

Review of Mathematical Background

<u>The Euclidean space $\mathbb{R}^n$ (cont.)</u>

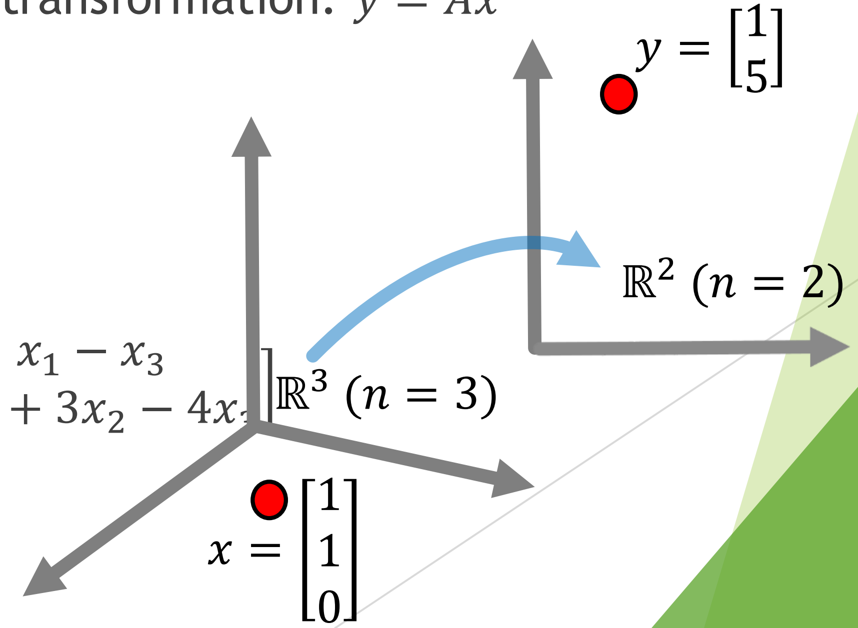
- Important types of linear combinations and their geometric:
  - **Linear combinations:** $\alpha x + \beta y$ where $\alpha, \beta \in \mathbb{R}$ span a 2D plane (linear sub-space)
  - **Affine combinations:** $\alpha x + \beta y$ where $\alpha + \beta = 1$ span a 1D line (affine subspace)
  - **Convex combinations:** $\alpha x + \beta y$ where $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$ span the line segment connecting x and y (the convex hull of x, y)
  - **Conic combinations:** $\alpha x + \beta y$ where and $\alpha, \beta \geq 0$ span the convex cone with apex at the origin and supported by x, y (we will better understand cones later in the course)

Review of Mathematical Background

Vectors and matrices

- Matrices are 2D arrays: $A = [a_{ij}]_{i=1,\dots,m, j=1,\dots,n}$ row index and column index
- Given bases for $\mathbb{R}^n, \mathbb{R}^m$, A represents a linear transformation: $y = Ax$
- For example: $A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 3 & -4 \end{bmatrix}$

$$y = Ax = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 - x_3 \\ 2x_1 + 3x_2 - 4x_3 \end{bmatrix}$$



Review of Mathematical Background

<u>Vectors and matrices</u>

- A matrix $A \in \mathbb{R}^{m \times n}$ induces four fundamental spaces:
  - The ***range (or image)*** of A: a subspace of $\mathbb{R}^m$: $R(A) \coloneqq \{Ax : x \in \mathbb{R}^n\}$
  - The ***null-space (or Kernel)*** of A: a subspace of $\mathbb{R}^n$: $N(A) \coloneqq \{x \in \mathbb{R}^n : Ax = 0\}$
  - The ***range (or image)*** of A^T: a subspace of $\mathbb{R}^n$: $R(A^T) \coloneqq \{A^T y : y \in \mathbb{R}^m\}$
  - The ***null-space (or Kernel)*** of A^T: a subspace of $\mathbb{R}^m$: $N(A^T) \coloneqq \{y \in \mathbb{R}^m : A^T y = 0\}$
- Take a few minutes to remember where each subspace is contained and which of them are the ***orthogonal complement*** of which

Review of Mathematical Background

<u>Vectors and matrices</u>

- Matrix vector multiplication of $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$:

$$Ax = y \in \mathbb{R}^m$$

- A linear transformation of x from $\mathbb{R}^n$ to $\mathbb{R}^m$
- Row view: each element (row) of y is a ***linear combination*** of the elements of x, with coefficients from the corresponding row of A

Review of Mathematical Background

Vectors and matrices (cont.)

- Matrix vector multiplication $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$:

$$Ax = y \in \mathbb{R}^m$$

- Column view: y is a ***linear combination of the columns*** of A, with coefficients given by x:

Denoting: $A = \begin{bmatrix} | & \cdots & | \\ a_1 & \dots & a_n \\ | & \cdots & | \end{bmatrix}$, then $y = x_1 a_1 + \cdots + x_n a_n$

Review of Mathematical Background

Vectors and matrices (cont)

- Five useful views of matrix multiplication $AB = C, A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times d}$
  - **View #1 - the definition** (rows × columns, scalar view): $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$
  - **View #2 - row view:** each row of C is a linear combination of the rows of B, with coefficients given by the respective row in A, i.e. row of A × the entire matrix B
  - **View #3 - column view:** each column of C is a linear combination of the columns of A, with coefficients given by the respective column in B, i.e. the entire matrix A × a column of B

Review of Mathematical Background

Vectors and matrices (cont)

- Five useful views of matrix multiplication $AB = C, A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times d}$
  - **View #4 - columns** $\times$ **rows** (rank-one view) C is the sum of n matrices, each given by column k of A $\times$ row k of B (each matrix of rank one)
  - **View #5 - block view:** the above can be done block-wise and still hold.

Review of Mathematical Background

<u>Eigenvalues and eigenvectors:</u>

- **Definition:** a nonzero vector v is an *eigenvector* of the square matrix A with *eigenvalue* λ (scalar) if $Av = \lambda v$
- Geometrically: directions in space where the matrix *operates as a scalar*
- If $A \in \mathbb{R}^{n \times n}$ admits n eigenvectors that are ***linearly independent***, denote by V the matrix with $v_1, \ldots, v_n$ stacked as columns, then we can write:

$$AV = \begin{bmatrix} | & & | \\ Av_1 & \cdots & Av_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \lambda_1 v_1 & \cdots & \lambda_n v_n \\ | & & | \end{bmatrix}$$

Review of Mathematical Background

Eigenvalues and eigenvectors (cont.):

- Now denote by Λ the diagonal matrix with elements $\lambda_1, \dots, \lambda_n$ in the diagonal, then:

$$AV = \begin{bmatrix} | & & | \\ Av_1 & \cdots & Av_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \lambda_1 v_1 & \cdots & \lambda_n v_n \\ | & & | \end{bmatrix} = V\Lambda$$

Hence: $AV = V\Lambda \Rightarrow \Lambda = V^{-1}AV$ (we got the diagonalization of A, also referred to as its *spectral decomposition*)

Review of Mathematical Background

<u>Symmetric Matrices</u>

- **Definition:** A is *symmetric* if $a_{ij} = a_{ji}$, i.e. $A = A^T$
- Properties of symmetric matrices:
  - All eigenvalues and eigenvectors are real
  - Eigenvectors are independent (hence diagonalizable)
  - Eigenvectors can be chosen such that they are *orthonormal*

Review of Mathematical Background

<u>**Orthogonal Matrices**</u>

- **Definition:** V is *orthogonal* if columns of V are orthonormal (each have unit length, and each pair is orthogonal)
- Properties of real an orthogonal matrix V:
  - $V^T V = I$, namely V^T is V^{-1}
  - $\det V = \pm 1$
  - Geometrically: rotations or reflections

Review of Mathematical Background

Orthogonal Matrices (cont)

Orthogonal matrices preserve distances (isometry):

$$\|Vx - Vy\|^2 = (Vx - Vy)^T(Vx - Vy) = (x^T V^T - y^T V^T)(Vx - Vy) = (x - y)^T V^T V(x - y)$$

$$= (x - y)^T(x - y) = \|x - y\|^2$$

Review of Mathematical Background

Orthogonal Matrices and Symmetric matrices

- Putting the above together, we have that a symmetric matrix A admits a spectral decomposition of the form: $\Lambda = V^T AV$ (with V orthogonal)
- (Remember that Hessian matrices of twice continuously differentiable functions are symmetric)

Review of Mathematical Background

Multi-variate derivatives:

- Denote by $f: \mathbb{R}^n \to \mathbb{R}$ a scalar valued function with continuous partial derivatives of second order

- The gradient vector is the vector of partial derivatives: $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$

Review of Mathematical Background

Multi-variate derivatives (cont.):

- By convention, the gradient is a column vector, i.e. a direction in the same space of the function's parameters
- When we refer to the *linear operator* we will denote it by df: the row vector (of partial derivatives) that operates by matrix multiplication
- More generally, for $F: \mathbb{R}^n \to \mathbb{R}^k$ denote by dF the matrix of partial derivatives: $[dF]_{ij} = \frac{\partial F_i}{\partial x_j}$, also called the *differential matrix* of F or the *Jacobian matrix*

Review of Mathematical Background

Multi-variate derivatives (cont.):

- Writing the derivatives in the above notation enables using the *chain rule* for vector valued functions
- Chain rule - function composition corresponds to matrix multiplication of the differential matrices:

$F: \mathbb{R}^n \to \mathbb{R}^k$ and $G: \mathbb{R}^k \to \mathbb{R}^d$ differentiable $\Rightarrow H \coloneqq G \circ F: \mathbb{R}^n \to \mathbb{R}^d$ differentiable and:

$$dH(x) = dG(y)dF(x)$$

Review of Mathematical Background

Multi-variate derivatives (cont.):

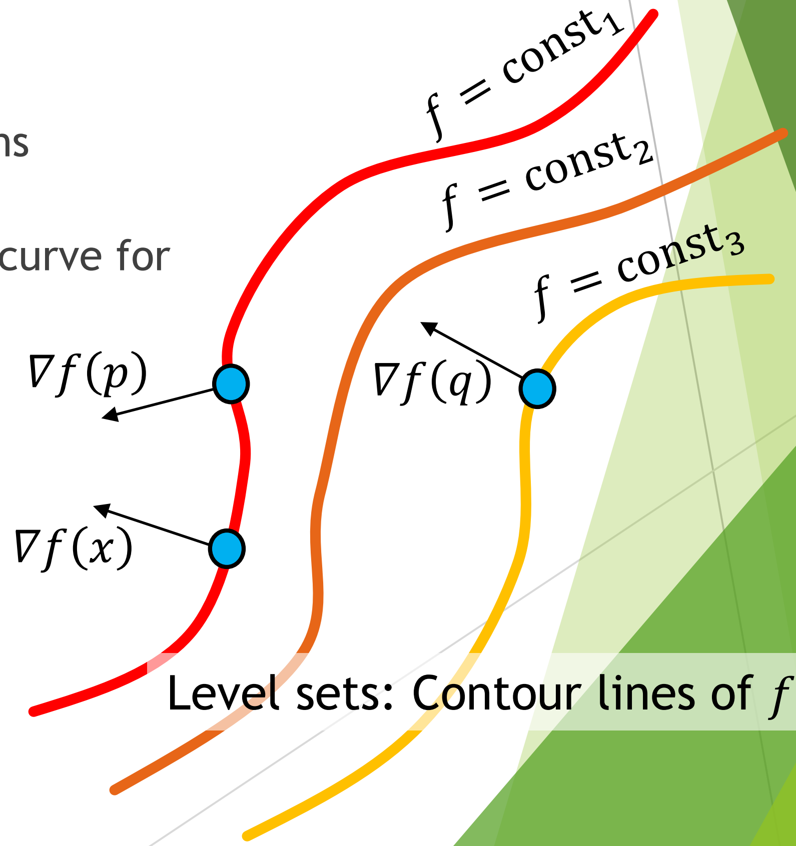
- Chain rule – example:

  gradients are orthogonal to level sets of differentiable functions

- Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable and $c(t),\ t \in (a,b)$ a smooth curve for which $f\big(c(t)\big) = \text{const}$

- Then $\frac{d}{dt}[f \circ c] = \frac{d}{dt}\text{const} = 0$

- But $\frac{d}{dt}[f \circ c](t) = df\big(c(t)\big)\frac{dc}{dt}(t) \Rightarrow$ the inner product: $\langle \nabla f, c'(t)\rangle$ is zero, and hence ∇f is orthogonal to tangent space



Level sets: Contour lines of f

Review of Mathematical Background

Multi-variate derivatives (cont.):

- Hessian: the matrix of second order derivatives of a scalar function
- Some important simple cases of vector differentiation (check by differentiation and make sure the look familiar from their univariate analogs!):
  - $f(x) = a^T x \Rightarrow \nabla f(x) = a$ (gradient of a linear function)
  - $f(x) = x^T Q x \Rightarrow \nabla f(x) = [Q + Q^T]x$ and if Q symmetric: $2Qx$ (gradient of a quadratic function)
  - $f(x) = x^T Q x \Rightarrow H(x) = 2Q$ (Hessian of a quadratic function)

Review of Mathematical Background

Multi-variate derivatives (cont.):

- Let u be a unit vector, that is: $u^T u = 1$
- For continuously differentiable functions, the *directional derivative* of f in the direction u (denoted $\partial_u f$) can be computed by the inner product: $\langle \nabla f, u \rangle$

Review of Mathematical Background

Multi-variate derivatives (cont.):

- Geometrically: the slope at direction u is in fact the projection of the gradient on u
- Therefore: $\partial_u f = \|\nabla f\|\|u\| \cos\theta$ and ∇f is the direction of greatest ascend, and $-\nabla f$ is the direction of steepest descent, which will play a role in several algorithms for minimization

Next Week

- Unconstrained optimization – problem definition
- Overview of algorithms for unconstrained optimization