Dynamic Programming

V. Leclère (ENPC)

February 24, 2023

Convention in these slides

Just a quick point about some unusual conventions I am using :

- ♡ means that the results in the slides are really important
- ♢ means that the content is more advanced
- ♣ is a very simple exercise (can be done in class)
- ♠ is a somewhat more difficult exercise that you can use as a training
- [BV x.y] means that the content is covered in the Convex Optimization book in chapter x, section y.

Why should I bother to learn this stuff?

- Markov Chains and Markov Decision Programms are very powerful modeling tools for a lot of practical applications.
- Dynamic programming is a flexible tool, easy to implement, that can efficiently address these problems.
- $\Longrightarrow$ useful for any "manager"

Contents

1. Controlled Markov Chain
2. Dynamic Programming
   - Markov Decision Problem
   - Dynamic Programming: Intuition
   - Dynamic Programming: Value function
   - Dynamic Programming: implementation
3. Infinite horizon
4. Parting thoughts
5. Wrap-up

Introduction ♡

- A Markov Chain $(\boldsymbol{X_t})_{t\in\mathbb{N}}$ is a *memoryless* stochastic process.
- A classical example is the random walk : let $(\boldsymbol{\xi_t})_{t\in\mathbb{N}}$ be a sequence of i.i.d. centered random variables and define

$$\boldsymbol{X_0} = 0, \qquad \boldsymbol{X_{t+1}} = \boldsymbol{X_t} + \boldsymbol{\xi_{t+1}}.$$

- A Markov chain can represent a large number of systems affected by random noises.
- A controlled Controlled Markov Chain is a Markov Chain such that the evolution is affected by an action.

Markov chain: definition ♢

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. Let $(\boldsymbol{X_t})_{t\in\mathbb{N}}$ be a sequence of discrete random variables taking value in $\mathcal{X}$. Let $\mathcal{F}_t = \sigma(\boldsymbol{X_0}, \dots, \boldsymbol{X_t})$ be the σ-algebra generated by all $\boldsymbol{X_\tau}$ for $\tau \leq t$.

We say that $(\boldsymbol{X_t})_{t\in\mathbb{N}}$ is a Markov Chain if

$$\mathbb{P}(\boldsymbol{X_t} \in A \mid \mathcal{F}_s) = \mathbb{P}(\boldsymbol{X_t} \in A \mid \boldsymbol{X_s}), \qquad \forall s \leq t, \forall A \text{ measurable}$$

or equivalently

$$\mathbb{E}[f(\boldsymbol{X_t}) \mid \mathcal{F}_s] = \mathbb{E}[f(\boldsymbol{X_t}) \mid \boldsymbol{X_s}], \qquad \forall s \leq t, \quad \forall f \quad \text{bounded and measurable}$$

If all $\boldsymbol{X_t}$ are discrete, this reads

$$\mathbb{P}(\boldsymbol{X_t} = x_t \mid \boldsymbol{X_0} = x_0, \dots, \boldsymbol{X_s} = x_s) = \mathbb{P}(\boldsymbol{X_t} = x_t \mid \boldsymbol{X_s} = x_s) \qquad \forall s \leq t, \forall x_0, \dots, x_t$$

Exercises

♣ Exercise: Show that if $(\boldsymbol{X_t})_{t\in\mathbb{N}}$ is a sequence of independent random variables then it is a Markov Chain.

♠ Exercise: Let $(\boldsymbol{\xi_t})_{t\in\mathbb{N}}$ be i.i.d. Assume that, for all $t \in \mathbb{N}$,

$$\boldsymbol{X_{t+k}} = \sum_{\kappa=0}^{k-1} \alpha_\kappa \boldsymbol{X_{t+\kappa}} + \boldsymbol{\xi_t}.$$

Show that $\boldsymbol{X_t}$ can easily be deduced from a Markov chain.

Discrete Markov chains ♡

Let $(\boldsymbol{X_t})_{t\in\mathbb{N}}$ be a Markov chain s.t. $\mathrm{supp}(\boldsymbol{X_t}) \subset \mathcal{X}$ where $\mathcal{X}$ is finite[1].

- We call $P_t : \mathcal{X}^2 \to [0, 1]$ the matrix such that,

$$P_t(x, y) = \mathbb{P}(\boldsymbol{X_{t+1}} = y | \boldsymbol{X_t} = x)$$

  the *t*-transition kernel of the Markov Chain $(\boldsymbol{X_t})_{t\in\mathbb{N}}$.
- A time-homogeneous Markov chain is such that $P_t = P$ for all t.

[1] extension to countable case are straightforward.

Chapman Kolmogorov equation ♢

- Let $\mu_t : \mathcal{X} \to [0,1]$ be a row vector such that representing the law of $\boldsymbol{X_t}$ (i.e $\mathbb{P}(X_t = x) = \mu(x)$), then we have (Chapman-Kolmogorov)[2]

$$\mu_{t+k} = \mu_t P^k.$$

- In particular, we have

$$\mathbb{P}(\boldsymbol{X_{t+k}} = y|\ \boldsymbol{X_t} = x) = P^k(x,y).$$

- Let $h : \mathcal{X} \to \mathbb{R}$, be represented as a column vector, then

$$P^k h(x) := \sum_{y \in \mathcal{X}} P^k(x,y)h(y) = \mathbb{E}[h(\boldsymbol{X_{t+k}})|\boldsymbol{X_t} = x].$$

[2] For simplicity the last three items are given under time-homogeneity.

Time homogeneous Markov chain graph representation

A simple way to represent a discrete Markov chain is through a directed graph:

- each node represents a state,
- we add an arc between node x and y iff $P(x,y) > 0$,
- when positive, we add the value $P(x,y)$ on the arc between x and y.

A time homogenous Markov chain is irreduccible if, starting from any point you can eventually reach any other points. More precisely, if for all $x, y \in \mathcal{X}$ there exists $t \in \mathbb{N}$ such that $\mathbb{P}(\boldsymbol{X_t} = y|\boldsymbol{X_0} = x) > 0$. Or equivalently if its graph is strongly connected.

Absorbing state

- An absorbing state of a Markov chain, is a state x such that there is no positive transition from x to another state $y \neq x$, that is such that $P(x,x) = 1$.
- If, from any state x there is a path to an absorbing state, then the Markov chain will almost surely end in an absorbing state.

Controlled Markov chains ♡

A controlled Markov chain is a Markov Chain whose transition kernel at time t is decided by an action $a_t \in \mathcal{A}$:

$$\mathbb{P}(\boldsymbol{X_{t+1}} = y|\boldsymbol{X_t} = x) = P_t^{a_t}(x,y).$$

- We consider a set of actions (or control) $\mathcal{A}$, assumed finite for simplicity.
- For all $t \in \mathbb{N}$ and $a \in \mathcal{A}$, let P_t^a be a transition kernel.
- We call a function π mapping the states $\mathcal{X}$ in to the action $\mathcal{A}$ a policy, and a collection $\pi = (\pi_t)_{t\in\mathbb{N}}$ a strategy.
- For any strategy π we define $(\boldsymbol{X_t^\pi})_{t\in\mathbb{N}}$ such that $(\boldsymbol{X_t}, \boldsymbol{a_t})_{t\in\mathbb{N}}$ is a Markov chain with

$$\mathbb{P}(\boldsymbol{X_{t+1}^\pi} = y, \boldsymbol{a_{t+1}} = b|\boldsymbol{X_t^\pi} = x, \boldsymbol{a_t} = a) = P_t^a(x,y)\mathbb{1}_{\pi_t(y)=b}.$$

Example and representation of Controlled Markov Chain

We consider a maintenance problem. A unit U can be either *working* or *broken*. When it is in a working state there is a 20% chance of being broken at the next time step. When it is broken it must be replaced and will be working at the next step.

♣ Exercise: Model this as a Markov Chain.

♣ Exercise: We now assume that at each time step, if the unit is working, we can decide to maintain it (keeping it in a working state) or not. And if broken we can repair it, or not. Model this modified version as a controlled Markov Chain.

Stochastic Dynamic System

- A (discrete time) stochastic dynamic system is a stochastic process $(\boldsymbol{X}_t)_{t\in\mathbb{N}}$ such that

$$\boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{a}_t, \boldsymbol{\xi}_t), \qquad \forall t$$

  where f_t is a deterministic function, $\boldsymbol{a}_t$ takes values in $\mathcal{A}$, and $\boldsymbol{\xi}_t$ is an exogenous random variable (i.e. its law is not affected by $\boldsymbol{X}_t$ and $\boldsymbol{a}_t$).
- All controlled Markov chains can be written as a stochastic dynamic system.
- If $(\boldsymbol{\xi}_t)_{t\in\mathbb{N}}$ is an independent sequence of random variables, then $(\boldsymbol{X}_t)_{t\in\mathbb{N}}$ is a controlled Markov chain.

Contents

1. Controlled Markov Chain
2. Dynamic Programming
   - Markov Decision Problem
   - Dynamic Programming: Intuition
   - Dynamic Programming: Value function
   - Dynamic Programming: implementation
3. Infinite horizon
4. Parting thoughts
5. Wrap-up

Markov Decision Problem ♡

- Let $(\boldsymbol{X}_t)_{t\in\mathbb{N}}$ be a controlled Markov chain, with action in $\mathcal{A}$. We denote Π the set of associated policies.
- Let, for all t, $c_t : \mathcal{X}^2 \to \mathbb{R} \cup +\infty$ be a transition cost.[3]
- A Markov Decision Problem is

$$\underset{\pi\in\Pi}{\text{Min}} \qquad \mathbb{E}\Big[\sum_{t\in\mathbb{N}} \rho^t c_t(\boldsymbol{X}_t^\pi, \boldsymbol{X}_{t+1}^\pi)\Big],$$

  where $\rho \in [0,1]$ is a discount factor.

[3] the transition cost can also be dependent on action a.

Another point of view

We can also write the MDP problem in the following way

$$\begin{array}{lll}\underset{(\pi_t)_{t\in\mathbb{N}}}{\text{Min}} & \mathbb{E}\Big[\mathbb{E}\Big[\sum_{t=1}^{\infty}\rho^t c_t(\boldsymbol{X_t},\boldsymbol{X_{t+1}}) \mid \boldsymbol{a_t}=\pi_t(\boldsymbol{X_t})\Big]\Big] & \\ \text{s.t.} & \boldsymbol{a_t}=\pi_t(\boldsymbol{X_t}) & \forall t\end{array}$$

Equivalently, with a stochastic dynamic system point of view, we have

$$\begin{array}{lll}\underset{(\pi_t)_{t\in\mathbb{N}}}{\text{Min}} & \mathbb{E}\Big[\sum_{t=1}^{\infty}\rho^t c_t(\boldsymbol{X_t},\boldsymbol{X_{t+1}})\Big] & \\ \text{s.t.} & \boldsymbol{a_t}=\pi_t(\boldsymbol{X_t}) & \forall t \\ & \boldsymbol{X_{t+1}}=f_t(\boldsymbol{X_t},\boldsymbol{a_t},\boldsymbol{\xi_t}) & \forall t\end{array}$$

Finite horizon problem

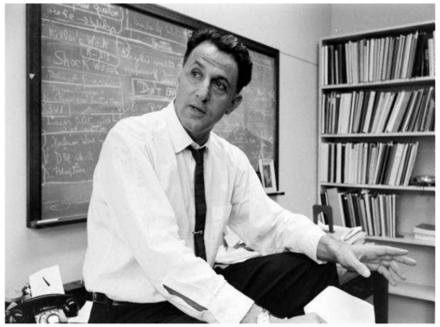
We now assume that for $t > T$, $c_t \equiv 0$, $\rho = 1$ and $c_T(x,y) = K(x)$. Thus the problem reads

$$\begin{array}{lll}\underset{(\pi_t)_{t\in\mathbb{N}}}{\text{Min}} & \mathbb{E}\Big[\sum_{t=1}^{T-1} c_t(\boldsymbol{X_t},\boldsymbol{X_{t+1}}) + K(\boldsymbol{X_T})\Big] & \\ \text{s.t.} & \boldsymbol{a_t}=\pi_t(\boldsymbol{X_t}) & \forall t \\ & \boldsymbol{X_{t+1}}=f_t(\boldsymbol{X_t},\boldsymbol{a_t},\boldsymbol{\xi_t}) & \forall t\end{array}$$

Further, we often assume that the initial state $\boldsymbol{X_0} = x_0$ is known.

This will be known as the Finite Horizon Problem.

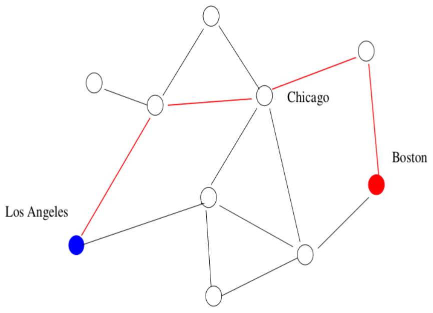
Bellman's Principle of Optimality



Richard Ernest Bellman (August 26, 1920 – March 19, 1984)

An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision (Richard Bellman)

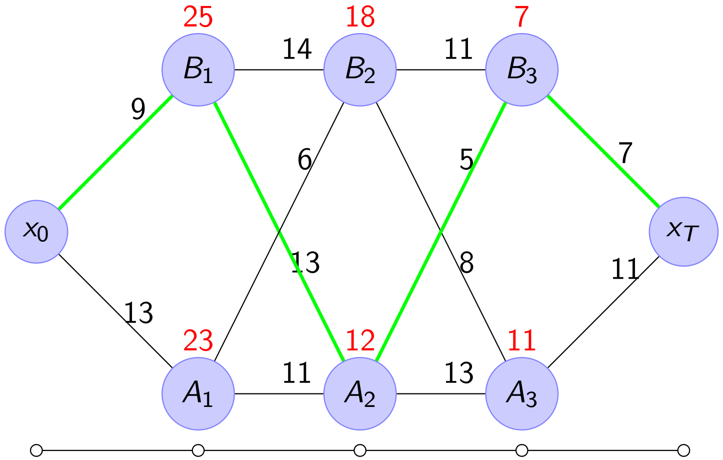
The shortest path on a graph illustrates Bellman's Principle of Optimality



For an auto travel analogy, suppose that the fastest route from Los Angeles to Boston passes through **Chicago**.
The principle of optimality translates to the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston. (Dimitri P. Bertsekas)

Idea Behind Dynamic Programming

Suppose that we have two states A and B and 4 timesteps.
For all t, we pay a cost to move from one node to one-another. We start from final position x_T, and computes cost to move from A_3 or B_3 to final position at time $t = 3$ We do the same at time $t = 2$, considering the cost to go to A_3 or B_3.



Idea behind dynamic programming

If we are in a Markovian setting, that is such that noises are time independent, then

1. The cost-to-go at time t depends only upon the current state.
2. We can compute recursively the cost to go for each position, starting from the terminal state and computing optimal trajectories backward.

Bellman's function: finite horizon ♡

In the finite horizon setting, the Bellman function reads

$$V_t(x) := \operatorname*{Min}_{\pi \in \Pi} \quad \mathbb{E}\Big[\sum_{\tau=t}^{T-1} c_\tau(\boldsymbol{X}_\tau^\pi, \boldsymbol{X}_{\tau+1}^\pi) + K(\boldsymbol{X}_T^\pi) \quad \big| \quad \boldsymbol{X}_t = x\Big]$$

and in particular $V_T = K$.
Or, in the stochastic dynamic system point of view

$$\begin{aligned} V_t(x) = \quad & \operatorname*{Min}_{(\pi_\tau)_{\tau \in [\![t, T-1]\!]}} \quad \mathbb{E}\Big[\sum_{\tau=t}^{T-1} c_\tau(\boldsymbol{X}_\tau, \boldsymbol{X}_{t+1}) + K(\boldsymbol{X}_T) \quad \big| \quad \boldsymbol{X}_t = x\Big] \\ & \text{s.t.} \quad \boldsymbol{a}_\tau = \pi_\tau(\boldsymbol{X}_\tau) \\ & \qquad \boldsymbol{X}_{t+1} = f_\tau(\boldsymbol{X}_\tau, \boldsymbol{a}_\tau, \boldsymbol{\xi}_\tau) \\ & \qquad \boldsymbol{X}_t = x \end{aligned}$$

Bellman's recursion : finite horizon ♡

In the finite horizon setting, we have

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min_{a \in \mathcal{A}} \mathbb{E}\Big[c_t(x, \boldsymbol{X}_{t+1}) + V_{t+1}(\boldsymbol{X}_{t+1}) \Big| \boldsymbol{X}_t = x, \ \boldsymbol{a}_t = a\Big] \\ & = \min_{a \in \mathcal{A}} \sum_{y \in \mathcal{X}} P^a(x, y)\Big(c_t(x, y) + V_{t+1}(y)\Big) \end{cases}$$

An optimal policy is given by

$$\pi_t(x) \in \arg\min_{a \in \mathcal{A}} \sum_{y \in \mathcal{X}} P^a(x, y)\Big(c_t(x, y) + V_{t+1}(y)\Big).$$

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal trajectory and value;
$V_T \equiv K$ ; for $t : T-1 \to 0$ do
    for $x \in \mathcal{X}$ do
        $V_t(x) = \infty$
        for $a \in \mathcal{A}$ do
            $Q(x,a) = 0$
            for $y \in \mathcal{X}$ do
                $Q(x,a) = Q(x,a) + P^a(x,y)[c_t(x,y) + V_{t+1}(y)]$
            if $Q(x,a) < V_t(x)$ then
                $V_t(x) = Q(x,a)$
                $\pi_t(x) = a$
```

Algorithm 1: Classical stochastic dynamic programming algorithm

3 curses of dimensionality ♡

Complexity $= O(T \times |\mathcal{X}|^2 \times |\mathcal{A}|)$
Linear in the number of time steps, but we have 3 curses of dimensionality:

1. State. Complexity is exponential in the dimension of $\mathcal{X}$
   e.g. 3 independent states each taking 10 values lead to a loop over 10^3 points.
2. Decision. Complexity is exponential in the dimension of $\mathcal{X}_t$.
   ⇝ due to exhaustive minimization of the inner problem. Can be accelerated using a faster method (e.g. MILP solver).
3. Expectation. Complexity is exponential in the dimension of Ξ_t.
   ⇝ due to expectation computation. Can be accelerated through Monte-Carlo approximation (still at least 1000 points)

In practice, DP is not used for state of dimension more than 5.

Some remarks ◇

- The loop on the next state y does not need to be on all state, but only on all *reachable* next state from state x.
- In some cases you do not need to compute the $V_t(x)$ for all $x \in \mathcal{X}$, indeed you might be able to show that some parts of the state space $\mathcal{X}$ are not reachable (or not reachable under an optimal policy) at time t.
- To represent that, at some time t, some state $x \in \mathcal{X}$ are forbidden, you can simply encode $V_t(x) = +\infty$.
- To represent that, at some time t, the transition $x \to y$ is forbidden, you can simply encode $c_t(x,y) = +\infty$.

Exercise

- Let $\mathcal{X} = \{0,1,2,3\}$, $\mathcal{A} = \{0,1\}$.
- Let $(\boldsymbol{X_t})_{t \in [\![1,5]\!]}$ be a controlled Markov chain, such that, if $a = 0$, it stays in its state, and if $a = 1$ it has a probability 0.5 of going 1 up (if possible, otherwise stay in place), and 0.5 of going 1 down (if possible, otherwise stay in place).

Solve by Dynamic Programming the following optimization problem.

$$\text{Max} \qquad \mathbb{E}\Big[\sum_{t=0}^{4} \boldsymbol{X_t}^2 \mid \boldsymbol{X_0} = 0\Big]$$

Contents

1 Controlled Markov Chain

2 Dynamic Programming
- Markov Decision Problem
- Dynamic Programming: Intuition
- Dynamic Programming: Value function
- Dynamic Programming: implementation

3 Infinite horizon

4 Parting thoughts

5 Wrap-up

Bellman's value function

The Bellman's value function, a.k.a cost-to-go function, is defined as (when the expectation make sense)

$$V_t(x) := \operatorname*{Min}_{\pi\in\Pi} \quad \mathbb{E}\Big[\sum_{\tau=t}^{+\infty} \rho^{\tau-t} c_\tau(\boldsymbol{X}^\pi_\tau, \boldsymbol{X}^\pi_{\tau+1}) \quad \Big| \quad \boldsymbol{X}_t = x\Big]$$

It is the value of the problem starting from time t in state x.

The expectation is well-defined for example if we consider a finite controlled Markov chain and one of the following holds:

- we are in the finite horizon framework,
- $c_t = c$ and $\rho < 1$,
- $c_t = c$ and there is a cemetery state (that is an aborbing state with null transition cost) that is almost surely reached.

Bellman's recursion

$$\begin{aligned}
V_t(x) &= \operatorname*{Min}_{\pi\in\Pi} \mathbb{E}\Big[\sum_{\tau=t}^{+\infty} \rho^{\tau-t} c_\tau(\boldsymbol{X}^\pi_\tau, \boldsymbol{X}^\pi_{\tau+1}) \Big| \boldsymbol{X}^\pi_t = x\Big] \\
&= \operatorname*{Min}_{\pi\in\Pi} \mathbb{E}\Big[c_\tau(\boldsymbol{X}^\pi_t, \boldsymbol{X}^\pi_{t+1}) + \sum_{\tau=t+1}^{+\infty} \rho^{\tau-t} c_\tau(\boldsymbol{X}^\pi_\tau, \boldsymbol{X}^\pi_{\tau+1}) \Big| \boldsymbol{X}^\pi_t = x\Big] \\
&= \operatorname*{Min}_{a\in\mathcal{A}} \sum_{y\in\mathcal{X}} P^a(x,y)\Big(c_t(x,y) + \operatorname*{Min}_{\pi\in\Pi} \mathbb{E}\Big[\sum_{\tau=t+1}^{+\infty} \rho^{\tau-t} c_\tau(\boldsymbol{X}^\pi_\tau, \boldsymbol{X}^\pi_{\tau+1}) \Big| \boldsymbol{X}^\pi_{t+1} = y\Big]\Big) \\
&= \operatorname*{Min}_{a\in\mathcal{A}} \sum_{y\in\mathcal{X}} P^a(x,y)(c_t(x,y) + \rho V_{t+1}(y)) \\
&= \operatorname*{Min}_{a\in\mathcal{A}} \mathbb{E}\Big[c_t(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}) + \rho V_{t+1}(\boldsymbol{X}_{t+1}) \Big| \boldsymbol{X}_t = x, \boldsymbol{a}_t = a\Big]
\end{aligned}$$

This equation should be understood as *the cost-to-go from state x and time t is equal to the minimum expected current cost plus futur cost.*

Stationary problem

From now on we make the following assumption:

- the set of possible values $\mathcal{X}$ is finite,
- the transition cost is not time dependent, i.e., $c_t = c$,
- the transition kernel is not time-dependent, i.e. $P^a_t = P^a$.

Then the MDP problem is said to be stationary.

A strategy $s = (\pi_t)_{t\in\mathbb{N}}$ is said to be stationary iff it is not time dependent, i.e. $\pi_t = \pi$.

A stationary MDP admits an optimal stationary policy.

Stochastic Shortest Path problem ◇

We consider a stationary MDP, with a cemetery state.

Stopping assumption

We assume that, for every state x, there exists T such that, under any (stationary) strategy π there is a positive probability of reaching the cemetery state.

♣ Exercise: Show that the finite horizon problem satisfies this stopping assumption.

♠ Exercise: Show that, if $\rho < 1$, even without an absorbing state, we can construct an equivalent MDP satisfying the stopping assumption.

Dynamic Programming equation ♡

Under this stopping assumption, the value function

$$V^\sharp(x) := \operatorname*{Min}_{\pi\in\Pi} \quad \mathbb{E}\Big[\sum_{\tau=0}^{+\infty} \rho^\tau c(\boldsymbol{X}^\pi_\tau, \boldsymbol{X}^\pi_{\tau+1}) \quad \Big]$$

Is the only function V satisfying the Dynamic Programming equation

$$\begin{cases} V(x) & = \min\limits_{a\in\mathcal{A}} \mathbb{E}\Big[c(x, \boldsymbol{X}_{t+1}) + \rho V(\boldsymbol{X}_{t+1}) \;\Big|\; \boldsymbol{X}_t = x,\ \boldsymbol{a}_t = a\Big] \\ & = \min\limits_{a\in\mathcal{A}} \sum\limits_{y\in\mathcal{X}} P^a(x,y)\Big(c(x,y) + \rho V(y)\Big) \end{cases}$$

Value iteration ♡

Define the following sequence of functions through the so-called Value Iteration procedure

$$\begin{cases} V_0 : x \mapsto 0 \\ V_{t+1} : x \mapsto \min\limits_{a\in\mathcal{A}} \sum\limits_{y\in\mathcal{X}} P^a(x,y)\Big(c_t(x,y) + \rho V_t(y)\Big) \end{cases}$$

Then we have, under the stopping assumption, $V_t \to V^\sharp$.

♠ Exercise: Recognize the Dynamic Programming algorithm of the finite horizon case. Interpret this result in terms of finite horizon approximation.

Optimal policy ◇

Naturally, a stationary policy π is optimal iff the minimum is attained in the DP equation, i.e.

$$V^\sharp(x) = \sum_{y\in\mathcal{X}} P^{\pi(x)}(x,y)\Big(c_t(x,y) + \rho V^\sharp(y)\Big), \qquad \forall x \in \mathcal{X}$$

Contents

1. Controlled Markov Chain
2. Dynamic Programming
   - Markov Decision Problem
   - Dynamic Programming: Intuition
   - Dynamic Programming: Value function
   - Dynamic Programming: implementation
3. Infinite horizon
4. Parting thoughts
5. Wrap-up

Law of large number and Central Limit Theorem ♡

Let $\{\boldsymbol{X}_i\}_{i\in\mathbb{N}}$ be a sequence of independent and identically distributed, real valued random variables. We denote the empirical mean $M_N = \frac{1}{N}\sum_{i=1}^N X_i$.

Theorem (LLN)

If $\boldsymbol{X}_1$ admits first order moment, then the empirical mean $\boldsymbol{M}_N$ converge almost surely toward the expectation $\mathbb{E}[\boldsymbol{X}_1]$.

Theorem (CLT)

If $\boldsymbol{X}_1$ admits second order moment, then we have

$$\sqrt{n}\Big(\boldsymbol{M}_N - \mathbb{E}\big[\boldsymbol{X}\big]\Big) \to \mathcal{N}(0,\sigma)$$

where the convergence is in law and σ is the standard deviation of $\boldsymbol{X}_1$.

In particular, the CLT means that, for $\boldsymbol{G} \sim \mathcal{N}(0,\sigma)$ and any $[a,b]$,

$$\mathbb{P}\Big(\sqrt{n}(\boldsymbol{M}_N - \mathbb{E}\big[\boldsymbol{X}\big]) \in [a,b]\Big) \to_N \mathbb{P}\Big(\boldsymbol{G} \in [a,b]\Big).$$

Monte-Carlo method ♡

- Let $\{\boldsymbol{X}_i\}_{i\in\mathbb{N}}$ be a sequence of rv iid with finite variance.
- We have $\mathbb{P}\Big(\boldsymbol{M}_N \in \Big[\mathbb{E}\big[\boldsymbol{X}\big] \pm \frac{\Phi^{-1}(1-p/2)std(\boldsymbol{X})}{\sqrt{N}}\Big]\Big) \approx p$
- In order to estimate the expectation $\mathbb{E}\big[\boldsymbol{X}\big]$, we can
  - sample N independent realizations of $\boldsymbol{X}$, $\{X_i\}_{i\in[\![1,N]\!]}$
  - compute the empirical mean $M_N = \frac{\sum_{i=1}^N X_i}{N}$, and standard-deviation s_N
  - choose an error level p (e.g. 5%) and compute $\Phi^{-1}(1-p/2)$ (1.96)
  - and we know that, asymptotically, the expectation $\mathbb{E}\big[\boldsymbol{X}\big]$ is in $\Big[M_N \pm \frac{\Phi^{-1}(p)s_N}{\sqrt{N}}\Big]$ with probability (on the sample) $1-p$

Good practice in optimization under uncertainty ♡

- Optimization under uncertainty is hard.
- You should first decide on a simulator for your problem, as precise as possible.
- Then, you should decide which problem you are going to solve. Most of the time it will be an approximation of the true problem.
- You can now solve, exactly or approximately this problem. Once you have a solution you should simulate it on your simulator (expected cost can be estimated by Monte Carlo).
- It is good practice to come up with reasonable *heuristic* to test your solution.

Contents

1. Controlled Markov Chain
2. Dynamic Programming
   - Markov Decision Problem
   - Dynamic Programming: Intuition
   - Dynamic Programming: Value function
   - Dynamic Programming: implementation
3. Infinite horizon
4. Parting thoughts
5. Wrap-up

What you have to know

- What is a Markov Chain, a Markov Controlled Chain.
- What is a Markov Decision Problem, a state, a policy
- What is the Bellman's value function a.k.a cost-to-go
- Estimate the value of a policy through Monte Carlo

What you really should know

- the complexity of Dynamic Programming
- how to model forbidden state in DP
- How to guarantee that an MDP in infinite horizon admits an optimal stationary policy

What you have to be able to do

- Recognize an MDP
- Write a Dynamic Programming equation
- Solve a simple, finite horizon, MDP problem through Dynamic Programming

What you should be able to do

- Know if a problem can numerically be tackled through Dynamic Programming
- Reframe a non-Markovian problem as a Markovian problem through extending the state
- Implement a value iteration algorithm in infinite horizon setting

Convexity

V. Leclère (ENPC)

March, 17th 2023

Why should I bother to learn this stuff?

- Convex vocabulary and results are needed throughout the course, especially to obtain optimality conditions and duality relations.
- Convex analysis tools like Fenchel transform appears in modern machine learning theory
- $\Longrightarrow$ fundamental for M2 in continuous optimization
- $\Longrightarrow$ usefull for M2 in operation research, machine learning (and some part of probability or mechanics)

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Affine sets ♡

Let X be a normed vector space (usually $X = \mathbb{R}^n$), and $C \subset X$

- C is affine if it contains any lines going through two distinct points of C, *i.e.*,
$$\forall x, y \in C, \quad \forall \theta \in \mathbb{R}, \qquad \theta x + (1-\theta) y \in C.$$
- The affine hull of C is the set of affine combination of elements of C,
$$\mathrm{aff}(C) := \Big\{ \sum_{i=1}^{K} \theta_i x_i \;\Big|\; \forall x_i \in C,\ \forall \theta_i \in \mathbb{R},\ \sum_{i=1}^{K} \theta_i = 1,\ \forall i \in [K], \forall K \in \mathbb{N} \Big\}$$
- $\mathrm{aff}(C)$ is the smallest affine space containing C.
- The affine dimension of C is the dimension of $\mathrm{aff}(C)$ (*i.e.*,the dimension of the vector space $\mathrm{aff}(C) - x_0$ for $x_0 \in C$).
- The relative interior of C is defined as
$$\mathrm{ri}(C) := \Big\{ x \in C \;\Big|\; \exists r > 0, \quad B(x,r) \cap \mathrm{aff}(C) \subset C \Big\}$$

Convex sets

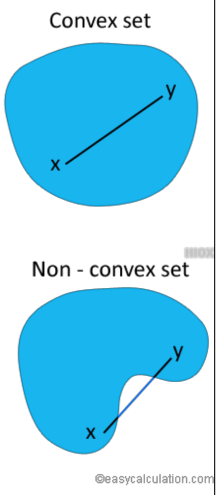
- C is convex if for any two points x and y in C the segment $[x, y] \subset C$, *i.e.*,

$$\forall x, y \in C,\ \forall \theta \in [0,1],\ \theta x + (1-\theta) y \in C.$$

- The convex hull of C as the set of convex combination of elements of C, *i.e.*,

$$\mathrm{conv}(C) := \Big\{ \sum_{i=1}^{K} \theta_i x_i \mid \ \forall x_i \in C,$$
$$\forall \theta_i \in [0,1],\ \sum_{i=1}^{K} \theta_i = 1,\ \forall i \in [K],\ \forall K \in \mathbb{N} \Big\}$$

- $\mathrm{conv}(C)$ is the smallest convex set containing C.


Convex set

Non - convex set

©easycalculation.com

Cones

- C is a cone if for all $x \in C$ the ray $\mathbb{R}_+ x \subset C$, *i.e.*,

$$\forall x \in C, \quad \forall \theta \in \mathbb{R}_+, \qquad \theta x \in C.$$

- The (convex) conic hull of C is the set of all (convex) conic combination of elements of C *i.e.*,

$$\mathrm{cone}(C) := \Big\{ \sum_{i=1}^{K} \theta_i x_i \mid \ \forall x_i \in C,\ \forall \theta_i \in \mathbb{R}_+,\ \forall i \in [K],\ \forall K \in \mathbb{N} \Big\}$$

- $\mathrm{cone}(C)$ is the smallest convex cone containing C.
- A cone C is pointed if it does not contain any full line $\mathbb{R}x$ for $x \neq 0$.
- For C convex, $\mathrm{cone}(C) = \bigcup_{t>0} tC$

Examples

Let $X = \mathbb{R}^n$.

- Any affine space is convex.
- Any hyperplane of X can be defined as $H := \{x \in X \mid \ a^\top x = b\}$ for well choosen $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ and is an affine space of dimension $n-1$.
- H divide X into two half-spaces $\{x \in \mathbb{R}^n \mid a^\top x \leq b$ and $\{x \in \mathbb{R}^n \mid a^\top x \geq b\}$ which are (closed) convex sets.
- For any norm $\| \cdot \|$ the ball $B_{\|\cdot\|}(x_0, r) := \{x \in X \mid \|x - x_0\| \leq r\}$ is a (closed) convex set.
  ♣ Exercise: Prove it.
- The set $C = \{(x, t) \in X \times \mathbb{R} \mid \|x\| \leq t\ \}$ is a cone.
- The set $C = \{x \in X \mid Ax \leq b\}$ where A and b are given is a (closed) convex set called polyhedron.

Operations preserving convexity

Assume that all sets denoted by C (indexed or not) are convex.

- $C_1 + C_2$ and $C_1 \times C_2$ are convex sets.
- For any arbitrary index set $\mathcal{I}$ the intersection $\bigcap_{i \in \mathcal{I}} C_i$ is convex.
- Let f be an affine function. Then $f(C)$ and $f^{-1}(C)$ are convex.
- In particular, $C + x_0$, and tC are convex. The projection of C on any affine space is convex.
- The closure $\mathrm{cl}(C)$ and relative interior $\mathrm{ri}(C)$ are convex.

♣ Exercise: Prove these results.

Perspective and linear-fractional function

Let $P : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ be the perspective function defined as $P(x,t) = x/t$, with $\mathrm{dom}(P) = \mathbb{R}^n \times \mathbb{R}^*_+$.

Theorem

If $C \subset \mathrm{dom}(P)$ is convex, then $P(C)$ is convex.
If $C \subset \mathbb{R}^n$ is convex, then $P^{-1}(C)$ is convex.

♠ Exercise: Prove this result.

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a linear-fractional function of the form $f(x) := (Ax+b)/(c^\top x + d)$, with $\mathrm{dom}(f) = \{x \mid c^\top x + d > 0\}$.

Theorem

If $C \subset \mathrm{dom}(f)$ is convex, then $f(C)$ and $f^{-1}(C)$ are convex.

♣ Exercise: prove this result.

Cone ordering

Let $K \subset \mathbb{R}^n$ be a closed, convex, pointed cone with non-empty interior. We define the cone ordering according to K by

$$x \preceq_K y \quad \Longleftrightarrow \quad y - x \in K.$$

♣ Exercise: Prove that $\preceq_K$ is a partial order (*i.e.*,reflexive, antisymmetric, transitive) compatible with scalar product, addition and limits.

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Separation

Let X be a Banach space, and X^* its topological dual (i.e. the set of all continuous linear forms on X).

Theorem (Simple separation)

Let A and B be convex non-empty, disjunct subsets of X. There exists a separating hyperplane $(x^*, \alpha) \in X^* \times \mathbb{R}$ such that

$$\langle x^*, a\rangle \leq \alpha \leq \langle x^*, b\rangle \qquad \forall a, b \in A \times B.$$

Theorem (Strong separation)

Let A and B be convex non-empty, disjunct subsets of X. Assume that, A is closed, and B is compact (e.g. a point), then there exists a strict separating hyperplane $(x^*, \alpha) \in X^* \times \mathbb{R}$ such that, there exists $\varepsilon > 0$,

$$\langle x^*, a\rangle + \varepsilon \leq \alpha \leq \langle x^*, b\rangle - \varepsilon \qquad \forall a, b \in A \times B.$$

Remark: these theorems require the Zorn Lemma which is equivalent to the axiom of choice.

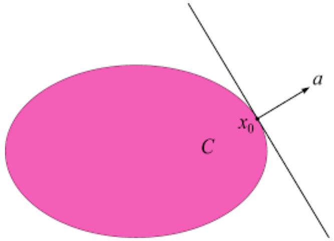
Supporting hyperplane

Theorem

Let $x_0 \notin \mathrm{ri}(C)$ and C convex. Then there exists $a \neq 0$ such that

$$a^\top x \geq a^\top x_0, \qquad \forall x \in C$$

If $x_0 \in C$, say that $H = \{x \mid a^\top x = a^\top x_0\}$ is a supporting hyperplane of C at x_0.



♣ Exercise: prove this theorem
Remark: there can be more than one supporting hyperplane at a given point.

Convex set as intersection of half-spaces

- The closed convex hull of $C \subset X$, denoted $\overline{\mathrm{conv}}(C)$ is the smallest closed convex set containing C.
- $\overline{\mathrm{conv}}(C)$ is the intersection of all the half-spaces containing C.
- A polyhedron is a finite intersection of half-spaces while a convex set is a possibly non-finite intersection of half-spaces.

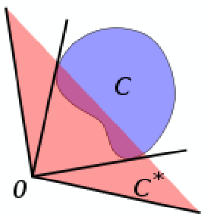
Dual and normal cones

- Let $C \subset \mathbb{R}^n$ be a set. We define its dual cone by

  $$C^\oplus := \{x \mid x^\top c \geq 0, \quad \forall c \in C\}$$

- For any set C, $C^\oplus$ is a closed convex cone.
- The normal cone of C at x_0 is

  $$N_C(x_0) := \{\lambda \in E \mid \lambda^\top (x - x_0) \leq 0, \quad \forall x \in C\}$$



Examples

- The positive orthant $K = \mathbb{R}^n_+$ is a self dual cone, that is $K^\oplus = K$.
- In the space of symetric matrices $S_n(\mathbb{R})$, with the scalar product $\langle A, B \rangle = \mathrm{tr}(AB)$, the set of positive semidefinite matrices $K = S_n^+(\mathbb{R})$ is self dual.
- Let $\|\cdot\|$ be a norm. The cone $K = \{(x,t) \mid \|x\| \leq t\}$ has for dual $K^\oplus = \{(\lambda, z) \mid \|\lambda\|_\star \leq z\}$, where $\|\lambda\|_\star := \sup_{x:\|x\| \leq 1} \lambda^\top x$.

♠ Exercise: prove these results

Some basic properties

Let $K \subset \mathbb{R}^n$ be a cone.

- $K^\oplus$ is closed convex.
- $K_1 \subset K_2$ implies $K_2^\oplus \subset K_1^\oplus$
- $K^{\oplus\oplus} = \overline{\mathrm{conv}}\, K$

♣ Exercise: Prove these results

Contents

1 Convex sets [BV 2]
- Fundamental definitions
- Separation theorems

2 Convex functions [BV 3]
- definitions
- Convex function and optimization
- Some results on convex functions

3 Convex analysis
- Subdifferential
- Fenchel transform

4 Wrap-up

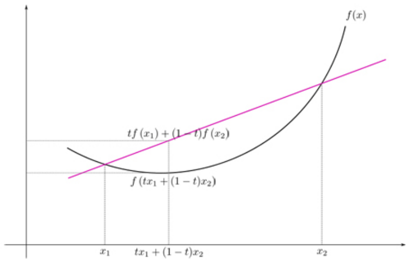
Functions with non finite values ♡

- It is very useful in optimization to allow functions to take non-finite values, that is to take values in $\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$.
- If both $-\infty$ and $+\infty$ are allowed be very careful of each addition !
- Let $f : X \to \bar{\mathbb{R}}$. We define
  - The epigraph of f as
$$\mathrm{epi}(f) := \{(x,t) \in X \times \mathbb{R} \quad | \quad f(x) \leq t\ \}$$
  - the domain of f as
$$\mathrm{dom}(f) := \{x \in X \quad | \quad f(x) < +\infty\}.$$
  - The sublevel set of level α
$$lev_\alpha(f) := \{x \in X \quad | \quad f(x) \leq \alpha\}.$$
- f is said to be lower semi continuous (l.s.c.) if $\mathrm{epi}(f)$ is closed.
- f is said to be proper if it never takes value $-\infty$, has a non-empty domain (at least one finite value).

Convex function ♡

- A function $f : X \to \bar{\mathbb{R}}$ is convex if its epigraph is convex.
- $f : X \to \mathbb{R} \cup \{+\infty\}$ is convex iff

$$\forall t \in [0,1],\ \forall x,y \in X,$$
$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$



- f is concave if $-f$ is convex.

Basic properties

- If f, g convex, $t > 0$, then $tf + g$ is convex.
- If f convex non-decreasing, g convex, then $f \circ g$ convex.
- If f convex and a affine, then $f \circ a$ is convex.
- If $(f_i)_{i \in I}$ is a family of convex functions, then $\sup_{i \in I} f_i$ is convex.
- The domain and the sublevel sets of a convex function are convex.
- A convex function is always above its tangents.

♣ Exercise: Prove these results.

Theorem (Jensen inequality)

Let f be a convex function and $\boldsymbol{X}$ an integrable random variable. Then we have

$$f(\mathbb{E}[\boldsymbol{X}]) \leq \mathbb{E}[f(\boldsymbol{X})].$$

Convex function : regularity

Consider a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$.

- f is continuous (on $\mathbb{R}^n$) if and only if $\mathrm{dom}(f) = \mathbb{R}^n$ (i.e., if it is finite everywhere)
- f is continuous on the interior of its domain
- f is lower-semicontinuous if and only if the domain is closed and the restriction of f to its domain is continuous

Convex functions: strict and strong convexity

- $f : X \to \mathbb{R} \cup \{+\infty\}$ is strictly convex iff

$$\forall t \in]0,1[, \quad \forall x, y \in X, \qquad f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$

- $f : X \to \mathbb{R} \cup \{+\infty\}$ is α-convex iff

$$\forall t \in]0,1[, \quad \forall x, y \in X, \qquad f(tx+(1-t)y) \leq tf(x)+(1-t)f(y)+\frac{1}{2}\alpha t(1-t)\|x-$$

- If $f \in C^1(\mathbb{R}^n)$
  - $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$ iff f convex
  - if strict inequality holds, then f strictly convex
  - $f : X \to \mathbb{R} \cup \{+\infty\}$ is α-convex iff $\forall x, y \in X$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2}\|y - x\|^2$$

- If $f \in C^2(\mathbb{R}^n)$,
  - $\nabla^2 f \succeq 0$ iff f convex
  - if $\nabla^2 f \succ 0$ then f strictly convex
  - if $\nabla^2 f \succeq \alpha I$ then f is α-convex

Important examples

- The indicator function of a set $C \subset X$,

$$\mathbb{I}_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \textit{otherwise} \end{cases}$$

  is convex iff C is convex.
- $x \mapsto e^{ax}$ is convex for any $a \in \mathbb{R}$
- $x \mapsto \|x\|^q$ is convex for $q \geq 1$ and any norm
- $x \mapsto \ln(x)$ is concave
- $x \mapsto x\ln(x)$ is convex
- $x \mapsto \ln(\sum_{i=1}^n e^{x_i})$ is convex

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Convex optimization problem ♡

$$\min_{x \in C} \quad f(x)$$

Where C is closed convex and f convex finite valued, is a convex optimization problem.

- If C is compact and f proper lsc, then there exists an optimal solution.
- If f is proper lsc and coercive, then there exists an optimal solution.
- The set of optimal solutions is convex.
- If f is strictly convex the minimum (if it exists) is unique.
- If f is α-convex the minimum exists and is unique.

♣ Exercise: Prove these results.

Optimality conditions ♡

Note that minimizing f over C or minimizing $f + \mathbb{I}_C$ over X is the same thing.
We consider the (unconstrained) optimization problem

$$\operatorname*{Min}_{x \in X} \qquad f(x),$$

with $x^\sharp$ an optimal solution and f not necessarily convex.

- If f is differentiable, then $\nabla f(x^\sharp) = 0$.
- If f is twice differentiable, then $\nabla^2 f(x^\sharp) \succeq 0$.
- If f is twice differentiable and $\nabla^2 f(x_0) \succ 0$ then x_0 is a local minimum.

If, in addition, f is convex then $\nabla f(x) = 0$ is a sufficient optimality condition.

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Partial infimum ♡

Let f be a convex function and C a convex set. The function

$$g : x \mapsto \inf_{y \in C} f(x, y)$$

is convex.

♠ Exercise: Prove this result.

♣ Exercise: Prove that the function distance to a convex set C defined by

$$d_C(x) := \inf_{c \in C} \|c - x\|$$

is convex.

Perspective function ♢

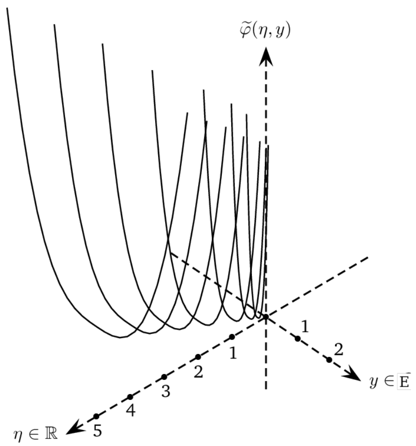
Let $\phi : E \to \bar{\mathbb{R}}$. The perspective of ϕ is defined as $\tilde{\phi} : \mathbb{R}_+^* \times E \to \mathbb{R}$ by

$$\tilde{\phi}(\eta, y) := \eta \phi(y/\eta).$$

Theorem

ϕ is convex iff $\tilde{\phi}$ is convex.

♠ Exercise: prove this result



Inf-Convolution ♢

Let f and g be proper function from X to $\mathbb{R} \cup \{+\infty\}$. We define

$$f \square g : x \mapsto \inf_{y \in X} f(y) + g(x - y)$$

♣ Exercise: Show that

- $f \square g = g \square f$
- If f and g are convex then so is $f \square g$

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Subdifferential of convex function

Let X be an Hilbert space, $f : X \to \bar{\mathbb{R}}$ convex.

- The subdifferential of f at $x \in \mathrm{dom}(f)$ is the set of slopes of all affine minorants of f exact at x:

$$\partial f(x) := \Big\{\lambda \in X \quad | \quad f(\cdot) \geq \langle \lambda, \cdot - x \rangle + f(x)\Big\}.$$

- If f is derivable at x then

$$\partial f(x) = \{\nabla f(x)\}.$$

Examples

- If $f : x \mapsto |x|$, then

$$\partial f(x) = \begin{cases} -1 & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

- If C is convex then, for $x \in C$, $\partial(\mathbb{I}_C)(x) = N_C(x)$
  ♣ Exercise: Prove it.
- If f_1 and f_2 are convex and differentiable. Define $f = \max(f_1, f_2)$. Then
  - if $f_1(x) > f_2(x)$, $\partial f(x) = \{\nabla f_1(x)\}$
  - if $f_1(x) < f_2(x)$, $\partial f(x) = \{\nabla f_2(x)\}$;
  - if $f_1(x) = f_2(x)$, $\partial f(x) = \overline{\mathrm{conv}}(\{\nabla f_1(x), \nabla f_2(x)\})$.

Subdifferential calculus

Let f_1 and f_2 be proper convex functions.

Theorem

We have

$$\partial(f_1)(x) + \partial(f_2)(x) \subset \partial(f_1 + f_2)(x), \qquad \forall x$$

Further if $\mathrm{ri}(\mathrm{dom}(f_1)) \cap \mathrm{ri}(\mathrm{dom}(f_2)) \neq \emptyset$ *then*

$$\partial(f_1)(x) + \partial(f_2)(x) = \partial(f_1 + f_2)(x), \qquad \forall x$$

When f_i is polyhedral you can replace $\mathrm{ri}(\mathrm{dom}(f_i))$ *by* $\mathrm{dom}(f_i)$ *in the condition.*

Theorem

If f is convex and $a : x \mapsto Ax + b$ with $Im(a) \cap \mathrm{ri}(\mathrm{dom}(f)) \neq \emptyset$, then

$$\partial(f \circ a)(x) = A^\top \partial f(Ax + b).$$

First order optimality conditions

Theorem

Let $f : X \mapsto \mathbb{R} \cup \{+\infty\}$ be a convex function (not necessarily) differentiable. $x^\sharp$ is a minimizer of f if and only if $0 \in \partial f(x^\sharp)$.

Theorem

Let f be a proper convex function and C a closed non-empty convex set such that $\mathrm{ri}(C) \cap \mathrm{ri}(\mathrm{dom}(f)) \neq \emptyset$ *then $x^\sharp$ is an optimal solution to*

$$\min_{x \in C} \qquad f(x)$$

iff

$$0 \in \partial f(x^\sharp) + N_C(x^\sharp),$$

iff

$$\exists \lambda \in \partial f(x^\sharp), \quad \lambda \in -N_C(x^\sharp).$$

Normal cone, Tangent cone and optimality

Let C be a convex set. We define the tangent cone of $C \subset \mathbb{R}^n$ at point $x \in C$, as the set of directions in which you can move from x while staying in C for some time, that is

$$T_C(x) := \left\{ \lambda(y-x) \;\middle|\; y \in C, \quad \lambda \in \mathbb{R}^+ \right\}$$

In particular, $T_C(x) = \mathbb{R}^n$ iff $x \in int(C)$.

♣ Exercise: Prove that $[T_C(x)]^\oplus = -N_C(x)$.

Partial infimum ♢

Let $f : X \times Y \to \bar{\mathbb{R}}$ be a jointly convex and proper function, and define

$$v(x) = \inf_{y \in Y} f(x,y)$$

then v is convex.
If v is proper, and $v(x) = f(x, y^\sharp(x))$ then

$$\partial v(x) = \{ g \in X \quad | \quad (g,0) \in \partial f(x, y^\sharp(x)) \}$$

proof:

$$\begin{aligned}
g \in \partial v(x) \quad &\Leftrightarrow \quad \forall x', \qquad v(x') \geq v(x) + \langle g, x' - x \rangle \\
&\Leftrightarrow \quad \forall x', y' \quad f(x',y') \geq f(x, y^\sharp(x)) + \left\langle \begin{pmatrix} g \\ 0 \end{pmatrix}, \begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} x \\ y^\sharp(x) \end{pmatrix} \right\rangle \\
&\Leftrightarrow \quad \begin{pmatrix} g \\ 0 \end{pmatrix} \in \partial f(x, y^\sharp(x))
\end{aligned}$$

Convex function: regularity ♢

- Assume f convex, then f is continuous on the relative interior of its domain, and Lipschitz on any compact contained in the relative interior of its domain.
- A proper convex function is subdifferentiable on the relative interior of its domain.
- If f is convex, it is L-Lipschitz iff $\partial f(x) \subset B(0,L), \quad \forall x \in \mathrm{dom}(f)$

Contents

1. Convex sets [BV 2]
   - Fundamental definitions
   - Separation theorems
2. Convex functions [BV 3]
   - definitions
   - Convex function and optimization
   - Some results on convex functions
3. Convex analysis
   - Subdifferential
   - Fenchel transform
4. Wrap-up

Fenchel transform

Let X be a Hilbert space, $f : X \to \bar{\mathbb{R}}$ be a proper function.

- The Fenchel transform of f, is $f^\star : X \to \bar{\mathbb{R}}$ with

$$f^\star(\lambda) := \sup_{x\in X} \langle \lambda, x\rangle - f(x).$$

- $f^\star$ is convex lsc as the supremum of affine functions.
- $f \leq g$ implies that $f^\star \geq g^\star$.
- If f is proper convex lsc, then $f^{\star\star} = f$, otherwise $f^{\star\star} \leq f$.

♣ Exercise: Prove the first two points

Fenchel transform and subdifferential

- By definition $f^\star(\lambda) \geq \langle \lambda, x\rangle - f(x)$ for all x,
- thus we always have (Fenchel-Young) $f(x) + f^\star(\lambda) \geq \langle \lambda, x\rangle$.
- Recall that $\lambda \in \partial f(x)$ iff for all x',

$$f(x') \geq f(x) + \langle \lambda, x' - x\rangle$$

iff

$$\langle \lambda, x\rangle - f(x) \geq \langle \lambda, x'\rangle - f(x') \qquad \forall x'$$

that is

$$\lambda \in \partial f(x) \Leftrightarrow x \in \arg\max_{x'\in X} \big\{\langle \lambda, x'\rangle - f(x')\big\} \Leftrightarrow f(x) + f^\star(\lambda) = \langle \lambda, x\rangle$$

- From Fenchel-Young equality we have

$$\partial v^{\star\star}(x) \neq \emptyset \quad \Longrightarrow \quad \partial v^{\star\star}(x) = \partial v(x) \text{ and } v^{\star\star}(x) = v(x).$$

- If f proper convex lsc

$$\lambda \in \partial f(x) \quad \Longleftrightarrow \quad x \in \partial f^\star(\lambda).$$

What you have to know

- What is a affine set, a convex set, a polyhedron, a (convex) cone
- What is a convex function, that it is above its tangents.
- Jensen inequality
- What is a convex optimization problem. That any local minimum is a global minimum.
- The necessary optimality condition $\nabla f(x^\sharp) \in [T_X(x^\sharp)]^\oplus$

What you really should know

- That you can separate convex sets with a linear function
- What is the positive dual of a cone
- Basic manipulations preserving convexity (sum, cartesian product, intersection, linear projection)
- What is the domain, the sublevel of a function f
- What is a lower semi-continuous function, a proper convex function
- Conditions of (strict, strong) convexity for differentiable functions
- The partial minimum of a convex function is convex
- The definition of the subdifferential.
- The definition of the Fenchel transform.
- The link between Fenchel transform and subdifferential.

What you have to be able to do

- Show that a set is convex
- Show that a function is (strictly, strongly) convex
- Go from constrained problem to unconstrained problem using the indicator function $\mathbb{I}_X$

What you should be able to do

- Compute dual cones
- Use advanced results (projection, partial infimum, perspective) to show that a function or a set is convex
- Compute the Fenchel transform of simple functions

Optimality conditions

V. Leclère (ENPC)

March 31st, 2023

Why should I bother to learn this stuff?

- Optimality conditions enable to solve exactly some easy optimization problems (e.g. in microeconomics, some mechanical problems...)
- Optimality conditions are used to derive algorithms for complex problem
- $\Longrightarrow$ fundamental both for studying optimization as well as other science

Contents

1. Optimization problem [BV 4.1]
2. Unconstrained case [BV 4.2]
3. First order optimality conditions [B.V 5.5]
4. Wrap-up

Optimization problem: vocabulary ♡

Generically speaking, an optimization problem is

$$\underset{x\in X}{\text{Min}} \qquad f(x) \qquad (P)$$

where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function (a.k.a. cost function),
- X is the feasible set,
- $x \in X$ is an admissible decision variables or a solution,
- $x^\sharp \in X$ such that $val(P) = f(x^\sharp) = \inf_{x\in X} f(x)$ is an optimal solution,
- if $X = \mathbb{R}^n$ the problem is unconstrained,
- if X and f are convex, then the problem is convex,
- if X is a polyhedron and f linear then the problem is linear,
- if X is a convex cone and f linear then the problem is conic.

Optimization problem: explicit formulation ♡

The previous optimization problem is often defined explicitly in the following standard form

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & f(x) && (P) \\ \text{s.t.} \quad & g_i(x) = 0 && \forall i \in [n_E] \\ & h_j(x) \leq 0 && \forall j \in [n_I] \end{aligned}$$

with

$$X := \{x \in \mathbb{R}^n \mid \forall i \in [n_E], \quad g_i(x) = 0, \quad \forall j \in [n_I], \quad h_j(x) \leq 0\}.$$

- (P) is a differentiable optimization problem if f and $\{g_i\}_{i\in[n_E]}$ and $\{h_j\}_{j\in[n_I]}$ are differentiable.
- (P) is a convex differentiable optimization problem if f, and h_j (for $j \in [n_I]$) are convex differentiable and g_i (for $i \in [n_E]$) are affine.
  ♣ Exercise: Show that in this case X is convex.

A few remarks and tricks ♢

- We can always write an abstract optimization problem in standard form (exercise!)
- For a given optimization problem there is an infinite number of possible standard forms (exercise!)
- We can always find an equivalent problem in dimension $\mathbb{R}^{n+1}$ with linear cost (exercise!)
- A minimization problem with $X = \emptyset$ has value $+\infty$ (by convention)
- A minimization problem has value $-\infty$ iff there exists a sequence $x_n \in X$ such that $f(x_n) \to -\infty$
- Maximizing f is just minimizing $-f$

Contents

1. Optimization problem [BV 4.1]
2. Unconstrained case [BV 4.2]
3. First order optimality conditions [B.V 5.5]
4. Wrap-up

Differentiable case ♡

Theorem

Assume that $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is differentiable at $x^\sharp$.

1. *If $x^\sharp$ is an unconstrained local minimizer of f then $\nabla f(x^\sharp) = 0$.*
2. *If in addition f is convex, then $\nabla f(x^\sharp) = 0$ iff $x^\sharp$ is a global minimizer.*

Proof:

1. Assume $\nabla f(x^\sharp) \neq 0$. DL of order 1 at $x^\sharp$ show that $f(x^\sharp - t\nabla f(x^\sharp)) < f(x^\sharp)$ for $t > 0$ small enough.
2. $f(y) \geq f(x^\sharp) + \langle \nabla f(x^\sharp), y - x^\sharp\rangle$.

Convex case ♡

Theorem

Consider $f : \mathbb{R}^n \to \bar{\mathbb{R}}$. Then $x^\sharp$ is a global minimum iff

$$0 \in \partial f(x^\sharp)$$

Theorem

Consider a proper convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, and X a closed convex set, such that $\mathrm{ri}(\mathrm{dom}(f)) \cap \mathrm{ri}(X) \neq \emptyset$.
Then $x^\sharp$ is a minimizer of f on X iff there exists $g \in \partial f(x^\sharp)$ such that $-g \in N_X(x^\sharp)$.

proof : The technical assumption ensures that $\partial(f + \mathbb{I}_X) = \partial f + \partial(\mathbb{I}_X)$.
As $\partial(\mathbb{I}_X) = N_X$, we have, $0 \in \partial(f + \mathbb{I}_X)(x^\sharp)$ iff there exists $g \in \partial f(x^\sharp)$ such that $-g \in N_X(x^\sharp)$.

Contents

1. Optimization problem [BV 4.1]
2. Unconstrained case [BV 4.2]
3. First order optimality conditions [B.V 5.5]
4. Wrap-up

Tangent cones ♢

For $f : \mathbb{R}^n \to \mathbb{R}$, we consider an optimization problem of the form

$$\underset{x \in X}{\mathrm{Min}} \qquad f(x).$$

Definition

We say that $d \in \mathbb{R}^n$ is tangent to X at $x \in X$ if there exists a sequence $x_k \in X$ converging to x and a sequence $t_k \searrow 0$ such that

$$d = \lim_k \frac{x_k - x}{t_k}.$$

Let $T_X(x)$ be the tangent cone of X at x, that is, the set of all tangent to X at x.

Equivalently,

$$T_X(x) = \{\, d \in \mathbb{R}^n \ \mid \exists t_k \searrow 0,\ \exists d_k \to d,\ x + t_k d_k \in X \}$$

Optimality conditions - differentiable case

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and the optimization problem

$$(P) \qquad \underset{x \in X}{\mathrm{Min}} \qquad f(x).$$

If $x^\sharp \notin \mathrm{int}(X)$ we do not necessarily need to have $\nabla f(x^\sharp) = 0$, indeed we just to have $\langle d, \nabla f(x^\sharp)\rangle \leq 0$ for all "admissible" direction d.

Theorem

Assume that f is differentiable at $x^\sharp$.

1. *If $x^\sharp$ is a local minimizer of (P) we have*

$$\nabla f(x^\sharp) \in \left[T_X(x^\sharp)\right]^\oplus. \qquad (*)$$

2. *If f and X are both convex, and $(*)$ holds, then $x^\sharp$ is an optimal solution of (P)*

♠ Exercise: Prove this result.

Convex case ♢

Let $K_X^{ad}(x)$ be the cone of admissible direction

$$K_X^{ad}(x) := \{t(y-x) \in \mathbb{R}^n \quad | \quad y \in X, \quad t \geq 0\}$$

Lemma

If $X \subset \mathbb{R}^n$ is convex, and $x \in X$, we have

$$T_X(x) = \overline{K_X^{ad}(x)}.$$

Recall that

$$T_X(x) = \{\ d \in \mathbb{R}^n \ | \exists t_k \searrow 0, \ \exists d_k \to d, \ x + t_k d_k \in X\}$$

♠ Exercise: Prove this lemma

Differentiable constraints ♢

We consider the following set of admissible solution

$$X = \Big\{x \in \mathbb{R}^n \quad | \quad g_i(x) = 0, \ i \in [n_E] \quad h_j(x) \leq 0, \ j \in [n_j]\Big\},$$

where g and h are differentiable functions.

Recall that the tangent cone is given by

$$T_X(x) = \{\ d \in \mathbb{R}^n \ | \exists t_k \searrow 0, \ \exists d_k \to d, \ g(x+t_k d_k) = 0, \ h(x+t_k d_k) \leq 0\}$$

We define the linearized tangent cone

$$\begin{aligned} T_X^\ell(x) := \{\ d \in \mathbb{R}^n \ | & \left\langle \nabla g_i(x) \, , d \right\rangle = 0, \ \forall i \in [n_E] \\ & \left\langle \nabla h_j(x) \, , d \right\rangle \leq 0, \ \forall j \in I_0(x)\} \end{aligned}$$

where

$$I_0(x) := \{j \in [n_I] \quad | \quad h_j(x) = 0\}.$$

Constraint qualifications ♢

We always have

$$T_X(x) \subset T_X^\ell(x).$$

♣ Exercise: Prove it.

We say that the constraints are qualified at x if

$$T_X(x) = T_X^\ell(x).$$

Sufficient qualification conditions ♡

Recall that g and h are assumed differentiable.
We denote the index set of active constraints at x

$$I_0(x) := \{i \in [n_I] \quad | \quad h_i(x) = 0\}.$$

The following conditions are sufficient qualification conditions at x:

1. g and h_i for $i \in I_0(x)$ are locally affine;
2. (Slater) g is affine, h_j are convex, and there exists x_S such that $g(x_S) = 0$ and $h_j(x_S) < 0$;
3. (Mangasarian-Fromowitz) For all $\alpha \in \mathbb{R}^{n_E}$ and $\beta \in \mathbb{R}_+^{n_I}$,

$$\sum_{i \in [n_E]} \alpha_i \nabla g_i(x) + \sum_{j \in I_0(x)} \beta_j \nabla h_j(x) = 0 \qquad \Longrightarrow \qquad \alpha = 0 \text{ and } \beta = 0$$

Expliciting the optimality condition I ♢

Under constraint qualification, the optimality condition reads

$$\nabla f(x) \in \big[T_X^\ell(x)\big]^\oplus$$

where

$$T_X^\ell(x) = \{\, d \in \mathbb{R}^n \mid \underbrace{\langle \nabla g_i(x) , d \rangle = 0, i \in [n_I] \quad \langle \nabla h_j(x) , d \rangle \leq 0, j \in I_0(x)\}}_{= A_x d \in C}.$$

with $A_x = \begin{pmatrix} ((\nabla g_i(x))^\top)_{i\in[n_I]} \\ ((\nabla h_j(x))^\top)_{j\in I_0(x)} \end{pmatrix}$ and $C = \{0\}^{n_E} \times (\mathbb{R}_-)^{n_I}$.

♣ Exercise: Show that $C^\oplus = \mathbb{R}^{n_E} \times (\mathbb{R}_-)^{n_I}$

Expliciting the optimality condition II ♢

Recall that the positive dual cone of a set K is

$$K^\oplus := \{d \in \mathbb{R}^n \mid \langle d, x \rangle \geq 0, \forall x \in K\}.$$

Let C be a closed convex set. Consider

$$K = A^{-1}C := \big\{x \in \mathbb{R}^n \mid Ax \in C\,\big\},$$

then

$$K^\oplus = \{\, A^\top \lambda \mid \lambda \in C^\oplus\}.$$

♣ Exercise: prove it.

Hence,

$$\nabla f(x) \in \big[\underbrace{T_X^\ell(x)}_{A_x^{-1}C}\big]^\oplus$$

$$\iff \qquad \exists \lambda \in C^\oplus, \quad \nabla f(x) = A_x^\top \lambda$$

$$\iff \quad \exists \lambda \in \mathbb{R}^{n_E},\ \exists \mu \in \mathbb{R}_+^{I_0(x)} \quad \nabla f(x) + \sum_{i=1}^{n_E} \lambda_i \nabla g_i(x) + \sum_{j \in I_0(x)} \mu_j \nabla h_j(x) = 0.$$

Karush Kuhn Tucker condition ♡

Theorem (KKT)

Assume that the objective function f and the constraint function g_i and h_j are differentiable. Assume that the constraints are qualified at x.

Then if x is a local minimum of

$$\min_{\tilde{x} \in \mathbb{R}^n} \Big\{ f(\tilde{x}) \quad \Big| \quad g_i(\tilde{x}) = 0,\ \forall i \in [n_E] \quad h_j(\tilde{x}) \leq 0,\ \forall j \in [n_I] \Big\}$$

then there exists dual variables λ, μ such that

$$\begin{cases} \nabla f(x) + \sum_{i=1}^{n_E} \lambda_i \nabla g_i(x) + \sum_{j=1}^{n_I} \mu_j \nabla h_j(x) = 0 & \nabla_x \mathcal{L} = 0 \\ g(x) = 0, \quad h(x) \leq 0 & \textit{Primal feasibility} \\ \lambda \in \mathbb{R}^{n_E}, \quad \mu \in \mathbb{R}_+^{n_I} & \textit{dual feasibility} \\ \mu_j h_j(x) = 0 \qquad \forall j \in [n_I] & \textit{complementarity constraint} \end{cases}$$

Contents

1. Optimization problem [BV 4.1]
2. Unconstrained case [BV 4.2]
3. First order optimality conditions [B.V 5.5]
4. Wrap-up

What you have to know

- Basic vocabulary: objective, constraint, admissible solution, differentiable optimization problem
- First order necessary KKT conditions

What you really should know

- What is a tangent cone
- Sufficient qualification conditions (linear and Slater's)
- That KKT conditions are sufficient in the convex case

What you have to be able to do

- Write the KKT condition for a given explicit problem and use them to solve said problem

What you should be able to do

- Check that constraints are qualified

Duality

V. Leclère (ENPC)

April 14th, 2023

Why should I bother to learn this stuff?

- Duality allow a second representation of the same convex problem, giving sometimes some interesting insights (e.g. principle of virtual forces in mechanics)
- Duality is a good way of obtaining lower bounds
- Duality is a powerful tool for decomposition methods
- $\Longrightarrow$ fundamental both for studying optimization (continuous and operations research)
- $\Longrightarrow$ usefull in other fields like mechanics and machine learning

Contents

1. Lagrangian duality [BV 5]
2. Strong duality
3. Marginal interpretation of the multiplier
4. Revisiting the KKT conditions
5. Wrap-up

Min-Max duality ♡

Consider the following problem

$$\underset{x\in\mathcal{X}}{\text{Min}} \quad \sup_{y\in\mathcal{Y}} \qquad \Phi(x,y)$$

where, for the moment, $\mathcal{X}$ and $\mathcal{Y}$ are arbitrary sets, and Φ an arbitrary function.
By definition the dual of this problem is

$$\underset{y\in\mathcal{Y}}{\text{Max}} \quad \inf_{x\in\mathcal{X}} \qquad \Phi(x,y)$$

and we have weak duality, that is

$$\sup_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} \Phi(x,y) \leq \inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} \Phi(x,y)$$

♣ Exercise: Prove this result.

Dual representation of some characteristic functions

Recall that, if $X \subset \mathbb{R}^n$

$$\mathbb{I}_X(x) = \begin{cases} 0 & \text{if } x \in X \\ +\infty & \text{otherwise} \end{cases}$$

and if X is an assertion,

$$\mathbb{I}_X = \begin{cases} 0 & \text{if } X \\ +\infty & \text{otherwise} \end{cases}$$

Note that

$$\mathbb{I}_{g(x)=0} = \sup_{\lambda \in \mathbb{R}^{n_E}} \lambda^\top g(x)$$

and

$$\mathbb{I}_{h(x)\leq 0} = \sup_{\mu \in \mathbb{R}^{n_I}_+} \mu^\top h(x)$$

From constrained to min-sup formulation

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & f(x) & & (P) \\ \text{s.t.} \quad & g_i(x) = 0 & & \forall i \in [n_E] \\ & h_j(x) \leq 0 & & \forall j \in [n_I] \end{aligned}$$

Is equivalent to

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \quad f(x) + \mathbb{I}_{g(x)=0} + \mathbb{I}_{h(x)\leq 0}$$

or

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \quad f(x) + \sup_{\lambda \in \mathbb{R}^{n_E}} \lambda^\top g(x) + \sup_{\mu \in \mathbb{R}^{n_I}_+} \mu^\top h(x)$$

which is usually written

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \quad \sup_{\lambda, \mu \geq 0} \quad \underbrace{f(x) + \lambda^\top g(x) + \mu^\top h(x)}_{:=\mathcal{L}(x;\lambda,\mu)}$$

Lagrangian duality

To a (primal) problem (no convexity or regularity assumptions here)

$$\begin{aligned} (P) \quad \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & f(x) & & \\ \text{s.t.} \quad & g_i(x) = 0 & & \forall i \in [n_E] \\ & h_j(x) \leq 0 & & \forall j \in [n_I] \end{aligned}$$

we associate the Lagrangian

$$\mathcal{L}(x; \lambda, \mu) := f(x) + \lambda^\top g(x) + \mu^\top h(x)$$

such that

$$(P) \qquad \underset{x \in \mathbb{R}^n}{\text{Min}} \quad \sup_{\lambda, \mu \geq 0} \quad \mathcal{L}(x; \lambda, \mu)$$

The dual problem is defined as

$$(D) \qquad \underset{\lambda, \mu \geq 0}{\text{Max}} \quad \inf_{x \in \mathbb{R}^n} \quad \mathcal{L}(x; \lambda, \mu)$$

Weak duality

By the min-max duality, we easily see that

$$\mathrm{val}(D) \leq \mathrm{val}(P).$$

Further any admissible dual multipliers $\lambda \in \mathbb{R}^{n_E}$ $\mu \in \mathbb{R}^{n_I}_+$ yields a lower bound:

$$g(\lambda, \mu) := \inf_{x \in \mathbb{R}^n} \mathcal{L}(x; \lambda, \mu) \leq \mathrm{val}(D) \leq \mathrm{val}(P)$$

Obviously, any admissible solution $x \in \mathbb{R}^n$ (i.e. such that $g(x) = 0$ and $h(x) \leq 0$), yields an upper bound

$$\mathrm{val}(P) \leq f(x) = \sup_{\lambda, \mu \geq 0} \mathcal{L}(x; \lambda, \mu)$$

Contents

1. Lagrangian duality [BV 5]
2. Strong duality
3. Marginal interpretation of the multiplier
4. Revisiting the KKT conditions
5. Wrap-up

Min-Max duality

Recall the generic primal problem of the form

$$p^\star := \underset{x\in\mathcal{X}}{\mathrm{Min}} \quad \sup_{y\in\mathcal{Y}} \qquad \Phi(x,y)$$

with associated dual

$$d^\star := \underset{y\in\mathcal{Y}}{\mathrm{Max}} \quad \inf_{x\in\mathcal{X}} \qquad \Phi(x,y).$$

Recall that the duality gap $p^\star - d^\star \geq 0$.
We say that we have strong duality if $d^\star = p^\star$.

Saddle point

Definition

Let $\Phi : \mathcal{X} \times \mathcal{Y} \to \bar{\mathbb{R}}$ be any function. $(x^\sharp, y^\sharp)$ is a (local) saddle point of Φ on $\mathcal{X} \times \mathcal{Y}$ if

- $x^\sharp$ is a (local) minimum of $x \mapsto \Phi(x, y^\sharp)$.
- $y^\sharp$ is a (local) maximum of $y \mapsto \Phi(x^\sharp, y)$.

If there exists a Saddle Point $(x^\sharp, y^\sharp)$ of Φ, then there is strong duality, $x^\sharp$ is an optimal primal solution and $y^\sharp$ an optimal dual solution, i.e.

$$p^\star = d^\star = \Phi(x^\sharp, y^\sharp).$$

Sufficient conditions for saddle point ♢

Theorem

If

- $\mathcal{X}$ *and* $\mathcal{Y}$ *are convex, one of them is compact*
- Φ *is continuous*
- $\Phi(\cdot, y)$ *is convex for all* $y \in \mathcal{Y}$
- $\Phi(x, \cdot)$ *is concave for all* $x \in \mathcal{X}$

then there exists a saddle point (i.e. we can exchange "Min" and "Max").

Slater's conditions for convex optimization ♡

Consider the following convex optimization problem

$$
\begin{aligned}
(P) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & f(x) \\
\text{s.t.} \quad & Ax = b \\
& h_j(x) \leq 0 \qquad\qquad \forall j \in [n_I]
\end{aligned}
$$

We say that a point x^s such that $Ax^s = b$, $x^s \in \mathrm{ri}(\mathrm{dom}(f))$, and $h_j(x^s) < 0$ for all $j \in [n_I]$, is a Slater's point.

Theorem

If (P) is convex (i.e. f and h_j are convex), and there exists a Slater's point then there is strong (Lagrangian) duality.

Further if (P) admits an optimal solution $x^\sharp$ then $\mathcal{L}$ admits a saddle point $(x^\sharp, \lambda^\sharp)$, and $\lambda^\sharp$ is an optimal solution to (D).

Contents

1 Lagrangian duality [BV 5]

2 Strong duality

3 Marginal interpretation of the multiplier

4 Revisiting the KKT conditions

5 Wrap-up

Pertubed problem ♢

We consider the following perturbed problem

$$
\begin{aligned}
v(p,q) = \qquad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & f(x) \\
\text{s.t.} \quad & g(x) = p \\
& h(x) \leq q
\end{aligned}
$$

In particular we have $v(0,0) = \mathrm{val}(P)$.
By duality,

$$
v(p,q) \geq d(p,q) = \sup_{\lambda,\mu\geq 0} \inf_x f(x) + \lambda^\top (g(x) - p) + \mu^\top (h(x) - q).
$$

In particular, d is convex as a supremum of convex functions.

Marginal interpretation of the dual multiplier ♡

Assume that (P) is convex, and satisfies the Slater's qualification condition. In particular $v(0,0) = d(0,0)$.
Let (λ, μ) be optimal multiplier of (P).
We have, for any $x_{p,q}$ admissible for the perturbed problem, that is such that $g(x_{p,q}) = p$, $h(x_{p,q}) \leq q$,

$$
\begin{aligned}
\mathrm{val}(P) = v(0,0) &= \inf_x f(x) + \lambda^\top g(x) + \mu^\top h(x) \\
&\leq f(x_{p,q}) + \lambda^\top g(x_{p,q}) + \mu^\top h(x_{p,q}) \\
&\leq f(x_{p,q}) + \lambda^\top p + \mu^\top q
\end{aligned}
$$

In particular we have,

$$
v(p,q) = \inf_{x_{p,q}} f(x_{p,q}) \geq v(0,0) - \lambda^\top p - \mu^\top q
$$

which reads

$$
-(\lambda,\mu) \in \partial v(0,0)
$$

Exercise

♣ Exercise: Consider the following problem, for $b \in \mathbb{R}$,

$$\begin{aligned} \underset{x\in\mathbb{R}}{\text{Min}} \quad & x^2 \\ \text{s.t.} \quad & x \leq b \end{aligned}$$

1. Does there exist an optimal multiplier?
2. Without solving the dual, give the optimal multiplier μ_b.

Contents

1. Lagrangian duality [BV 5]
2. Strong duality
3. Marginal interpretation of the multiplier
4. Revisiting the KKT conditions
5. Wrap-up

KKT conditions ♡

Recall the first order KKT conditions for our problem (P)

$$\begin{aligned} & \nabla f(x) + \lambda^\top A + \sum_{j=1}^{n_I} \mu_j \nabla h_j(x) = 0 \\ & Ax = b, \quad h(x) \leq 0 \\ & \lambda \in \mathbb{R}^{n_E}, \quad \mu \in \mathbb{R}^{n_I}_+ \\ & \lambda_j g_j(x) = 0 \qquad\qquad \forall j \in [n_I] \end{aligned}$$

Further, recall that

- the existence of a Slater's point in a convex problem ensures constraints qualifications,
- first order conditions are sufficient for convex problems.

KKT and duality ♢

If (P) is convex and there exists a Slater's point. Then the following assertions are equivalent:

1. $x^\sharp$ is an optimal solution of (P),
2. $(\exists \lambda^\sharp$ such that$)$ $(x^\sharp, \lambda^\sharp)$ is a saddle point of $\mathcal{L}$,
3. $(\exists \lambda^\sharp$ such that$)$ $(x^\sharp, \lambda^\sharp)$ satisfies the KKT conditions.

Recovering KKT conditions from Lagrangian duality ♢

$$
\begin{array}{lll}
(P) & \underset{x\in\mathbb{R}^n}{\text{Min}} & f(x) \\
 & \text{s.t.} & A(x) = b \\
 & & h_j(x) \leq 0 \qquad\qquad \forall j \in [n_I]
\end{array}
$$

with associated Lagrangian

$$\mathcal{L}(x;\lambda,\mu) := f(x) + \lambda^\top\big(A(x)-b\big) + \mu^\top h(x)$$

The KKT conditions can be seen as:

1. $\nabla_x\mathcal{L}(x;\lambda,\mu) = 0$ (Lagrangian minimized in x)
2. $g(x) = 0$, $h(x) \leq 0$ (x primal admissible, also obtained as $\nabla_\lambda\mathcal{L} = 0$)
3. $\mu \geq 0$ ((λ,μ) dual admissible)
4. $\mu_j = 0$ or $h_j(x) = 0$, for all $j \in [n_I]$ (complementarity constraint $\rightsquigarrow 2^{n_I}$ possibilities).

Complementarity condition and marginal value interpretation ♢

Consider a convex problem satisfying Slater's condition.
Recall that $-\mu^\sharp \in \partial v(0)$ where $v(p)$ is the value of the perturbed problem.
From this interpretation, we can recover the complementarity condition

$$\mu_j = 0 \qquad \text{or} \qquad g_j(x) = 0$$

Indeed, let x be an optimal solution.

- If constraint j is not saturated at x (i.e $g_i(x) < 0$), we can marginally move the constraint without affecting the optimal solution, and thus the optimal value. In particular, it means that $\mu_j = 0$.
- If $\mu_j \neq 0$, it means that marginally moving the constraint changes the optimal value and thus the optimal solution. In particular, constraint j must be saturated, i.e $g_i(x) = 0$.

Contents

1. Lagrangian duality [BV 5]
2. Strong duality
3. Marginal interpretation of the multiplier
4. Revisiting the KKT conditions
5. Wrap-up

What you have to know

- Weak duality: $\sup\inf\Phi \leq \inf\sup\Phi$
- Definition of the Lagrangian $\mathcal{L}$
- Definition of primal and dual problem

$$\underbrace{\underset{\lambda,\mu}{\text{Max}}\ \inf_{x}\ \mathcal{L}(x;\lambda,\mu)}_{\text{Dual}} \leq \underbrace{\underset{x}{\text{Min}}\ \sup_{\lambda,\mu}\ \mathcal{L}(x;\lambda,\mu)}_{\text{Primal}}$$

- Marginal interpretation of the optimal multipliers

What you really should know

- A saddle point of $\mathcal{L}$ is a primal-dual optimal pair
- Sufficient condition of strong duality under convexity (Slater's)

What you have to be able to do

- Turn a constrained optimization problem into an unconstrained Min sup problem through the Lagrangian
- Write the dual of a given problem
- Heuristically recover the KKT conditions from the Lagrangian of a problem

What you should be able to do

- Get lower bounds through duality

Optimization and algorithms

V. Leclère (ENPC)

April 21th, 2023

Why should I bother to learn this stuff?

- Being able to recognize the type of problem is the first step toward finding the right tool to adress it.
- Having an idea of the tools available to you will help choose one.
- $\Longrightarrow$ usefull for any engineer (or intern) that might have to model and then solve a practical optimization problem.

Contents

1. Classification of optimization problems
   - Generic ideas
   - Some important classes
2. Optimization methods
   - Heuristics
   - Descent direction method
   - Model based methods
3. Wrap-up

Why bother with classes of optimization problems ?

Consider a function $f : X \to \mathbb{R}$, and the following optimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Solving this problem can be more or less hard depending on the class in which f and $X \subset \mathbb{R}^n$[1] belongs.

Determining in which class a problem belongs is quite important:

- some problem can be solved for n of order 10 at most, other for n of order 10^6 or more;
- the methodological approach to tackle different problems vary wildly;
- the numerical tools (e.g. solvers) also...

You must be able to (roughly) classify correctly the problem you face, in order to know what can be done or not.

[1] There is also an important theory of optimization where X is not contained in a finite-dimensional space, which will not be discussed here.

Classification with respect to the objective function f ♡

- f linear is the simplest case
- f quadratic is a very important case, simple if f is convex
- f smooth (e.g. $\mathcal{C}^2$) allow to use first and second order information on f
- (f polynomial is a special case, with specific algorithms)
- f convex imply that any local minimum is a global minimum

Finding the optimal solution is a reasonable goal only in the convex case. Otherwise, the algorithm aims at finding one or multiple local optima.

The algorithms we present are mainly for smooth functions. Convergence theory will be done in the convex case.

Classification with respect to the constraint set ♡

- $X = \mathbb{R}^n$, is known as unconstrained optimization.
- $X = \{x \in \mathbb{R}^n \mid Ax = b\}$, can be cast, up to reparametrization, as unconstrained optimization. It might be more efficient to directly deal with the constraints.
- $X = \{x \in \mathbb{R}^n \mid \underline{x}_i \leq x_i \leq \bar{x}_i, \quad \forall i \in [n]\}$ is the box constrained optimization.
- $X = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a polyhedron.
- X convex, generally given as $\{x \in \mathbb{R}^n \mid Ax = b, \quad h_j(x) \leq 0, \quad \forall j \in [n_I]\}$ with h_j convex.
- If X is a finite set we speak of combinatorial optimization.
- X can also be non-convex but smooth (e.g. a manifold)

A few comments:

- Unconstrained optimization is by far easier.
- Box constraints, and sometimes spherical constraints, are easy.
- Polyhedral constraints indicate LP-based methods.
- Integrity constraints make the problem a lot harder and change the nature of the optimization methods.

Contents

1. Classification of optimization problems
   - Generic ideas
   - Some important classes
2. Optimization methods
   - Heuristics
   - Descent direction method
   - Model based methods
3. Wrap-up

Least-square problem (LS)

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \qquad \|Ax - b\|_2$$

- equivalent (in which sense ?) to $\underset{x \in \mathbb{R}^n}{\text{Min}} \ \|Ax - b\|_2^2$
- $\rightsquigarrow$ convex, smooth, unconstrained problem
- explicit solution known through algebraic manipulation
- sometimes easier to solve by optimization method than algebraic manipulation
- can be (approximately) solved for $n \geq 10^{11}$ (sparse case)

Exercise : functional approximation

♣ Exercise: We consider a physical function Φ that is approximated as the superposition of multiple simple phenomena (e.g. waves). Each simple phenomenon $p \in [P]$ is represented by a function $\Phi_p : \mathbb{R}^d \to \mathbb{R}$.

We have data points $(x^k, y^k)_{k\in[n]}$, and want to find the Φ that match *at best* the data while being a linear combination of Φ_p.

Propose a least-square regression that answers this question.

Linear optimization problem (LP) ♡

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & A'x \leq b' \end{aligned}$$

- convex problem with linear objective and polyhedral constraint set
- a rare case where exact solution can be obtained
- easily solved through dedicated code, open-source (e.g. GLPK) or proprietary[2] (e.g. CPLEX, Gurobi)
- can be solved for $n \geq 10^8$
- very important case in practice and as a subroutine for other problems
- two main (class of) algorithms:
  - simplex algorithm (seen in 1A)
  - interior point method (discussed later in this course)

[2] Licences are expensive(!!) but free for students!

Quadratic optimization problem (QP) ♡

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & \frac{1}{2}x^\top Qx + c^\top x \\ \text{s.t.} \quad & Ax = b \\ & A'x \leq b' \end{aligned}$$

- quadratic objective and polyhedral constraint set
- exact solution can be obtained
- easily solved if $Q \succeq 0$, hard otherwise
- can be solved for $n \geq 10^7$ (convex case)

Exercise : Lasso as QP

♣ Exercise: A classical extension of the least-square problem, which has strong theoretical and practical interest is the LASSO problem

$$\underset{x\in\mathbb{R}^p}{\text{Min}} \quad \|Ax - y\|^2 + \lambda\|x\|_1$$

Show that this problem can be cast as a QP problem.

Quadratically constrained quadratic problem (QCQP)

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & \frac{1}{2}x^\top Qx + c^\top x \\ \text{s.t.} \quad & \frac{1}{2}x^\top P_i x + q_i^\top x \le b_i & \forall i \in [k] \\ & Ax = b \\ & A'x \le b' \end{aligned}$$

- Reasonably easy if convex (i.e if Q and P_i are semi-definite positive)
- can be solved for $n \ge 10^7$
- less important than previous examples

Exercise: binary optimization is equivalent to QCQP

♣ Exercise: Consider the following optimization problem.

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x_i \in \{0,1\} & \forall i \in I \end{aligned}$$

Write this problem as a QCQP. Is it convex ?

Second order cone problem (SOCP)

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & \|A_i x + b_i\|_2 \le c_i^\top x + d_i & \forall i \in [k] \\ & Ax = b \\ & A'x \le b' \end{aligned}$$

- convex problem
- can be solved for $n \ge 10^7$, through most "linear" solver, relying on interior points methods
- equivalent to convex QCQP
- extend the modeling power of LP
- appears naturally in robust optimization

Exercise : robust linear programming

♠ Exercise: Consider the following robust linear program

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & (a_i + R_i\delta_i)^\top x \le b_i & \forall \|\delta_i\|_2 \le 1, \quad \forall i \in [m] \end{aligned}$$

Write this problem as a SOCP.

Semi definite programming (SDP) ♢

$$\begin{aligned} \underset{X \in S_n(\mathbb{R})}{\text{Min}} \quad & \operatorname{tr}\Big(CX\Big) \\ \text{s.t.} \quad & A(X) = b \\ & X \succeq 0 \end{aligned}$$

where X, and C are symmetric matrices, and $A : S_n \to \mathbb{R}^m$ a linear mapping.

- convex problem
- can be solved for $n \geq 10^3$, through some "linear" solver, relying on interior points methods
- contains SOCP
- limited in size in part because the number of actual variables is n^2

Unconstrained convex-differentiable optimization ♡

$$\underset{x}{\text{Min}} \quad f(x)$$

where f is convex, finite, and differentiable.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Convergence speed is better under strong convexity assumptions
- Can be solved for $n \geq 10^5$

$\leadsto$ this is where we will spend most of our time

Unconstrained convex non-differentiable optimization

$$\underset{x}{\text{Min}} \quad f(x)$$

where f is convex and finite.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Can be solved for $n \geq 10^4$

Unconstrained differentiable optimization

$$\underset{x}{\text{Min}} \quad f(x)$$

where f is differentiable.

- Iterative algorithm yields ε local optimum
- Algorithms are mostly the same as in convex differentiable setting, but the theory is more involved
- Can find a local optimum for $n \geq 10^5$

$\leadsto$ most algorithms presented in this course can be used to hopefully get to a locally optimal point.

Constrained convex optimization

$$\begin{aligned} \underset{x}{\text{Min}} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

where X is a convex set.

- Easiest if X is a box or ball
- Specific approach relying on LP if X is a polyhedron
- Various methods in the generic case:
  - projection
  - feasible direction
  - constraint penalization
  - dualization

Combinatorial problem

$$\min_{x \in X} \quad f(x)$$

where X is a finite set.

- This roughly represent the problem of combinatorial optimization.
- X being finite you can, in theory, test all possibilities and choose the best. However, this brute force approach is often unpractical due to the size of X.
- Even if an exact solution can be obtained, it is not often the case.
- Finding lower-bound is interesting to understand how far your current solution is from the optimum.
- Practical methods are often *matheuristics* or *meta-heuristics* adapted to the specificity of the problem.
- Problems are often very hard, and practical solvability depends on the specific problem structure.

Mixed Integer Linear Programming ♡

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b, x \geq 0 \\ & x_i \in \mathbb{N} & \forall i \in I \subset [n] \end{aligned}$$

- A very important class of problem, with huge modeling power.
- By order of difficulty we distinguish: continuous variables, binary variables and integer variables.
- Exact solution methods rely on the idea of branch and cut. (`https://www.youtube.com/watch?v=2zKCQ03JzOY` (13'))
- Very powerful (commercial) solvers (like Gurobi, Cplex, Mosek...) are developed and improved every year to tackle these problems. They use a mix of insightful mathematical ideas and heuristic knowledge.
- Efficiency of the solver depends on the type of problem, and the formulation of the problem.

Mixed Integer Conic Programming ♢

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in C \\ & x_i \in \mathbb{N} & \forall i \in I \subset [n] \end{aligned}$$

where C is a convex cone.

- Harder than MILP
- More recent development, thus the theory and heuristic experience is less advanced than MILP
- Numerical efficiency is quickly improving

Exercise : stock optimization

♠ Exercise: Consider that you sell a given product over T days. The demand for each day is d_t. Having a quantity x_t of items in stock have a cost (per day) of cx_t. You can order, each day, a quantity q_t, and have to satisfy the demand.

For each of the following variations: model the problem, give the class to which it belongs, and give the optimal solution if easily found.

1. Without any further constraints / specifications.
2. There is an "ordering cost": each time you order, you have to pay a fixed cost κ.
3. Instead of an "ordering cost" there is a maximum number of days at which you can order a replenishment.

Contents

1. Classification of optimization problems
   - Generic ideas
   - Some important classes
2. Optimization methods
   - Heuristics
   - Descent direction method
   - Model based methods
3. Wrap-up

"Ad Hoc" solution

A heuristic is an admissible, not necessarily optimal, solution to a given optimization problem. It gives upper-bounds.

- In a lot of applications, experience or good sense, can give reasonably good heuristics.
- Sometimes these heuristics can have a few parameters that can be tuned by trial and error.

♣ Exercise: In the stock optimization example, with fixed ordering cost, propose a simple heuristic.

♣ Exercise: Now assume that, in this same example, there is some uncertainty on the demand, adapt your heuristic to offer a *robustness* parameter.

Random search

A good way of obtaining *good* solutions is to randomly test multiple admissible solutions, and keep the best one.

Examples:

- exhaustive search (combinatorial)
- genetic algorithms
- simulated annealing
- swarm particles

Use case :

- hard problems (combinatorial or continuous) where finding an admissible solution is easy
- when you just want an admissible solution, if possible better than what you had

Contents

1. Classification of optimization problems
   - Generic ideas
   - Some important classes
2. Optimization methods
   - Heuristics
   - Descent direction method
   - Model based methods
3. Wrap-up

Descent methods ♡

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{2}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

$\rightsquigarrow$ most of this is discussed in next classes.

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} \leq 0$, which can be seen from a first order development:

$$f(x^{(k)} + t^{(k)} d^{(k)}) = f(x^{(k)}) + t \big\langle \nabla f(x^{(k)}) \, , d^{(k)} \big\rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which correspond to the gradient algorithm.

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f(x^{(k)} + t d^{(k)})$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or linear search).

Contents

1. Classification of optimization problems
   - Generic ideas
   - Some important classes
2. Optimization methods
   - Heuristics
   - Descent direction method
   - Model based methods
3. Wrap-up

Model based methods

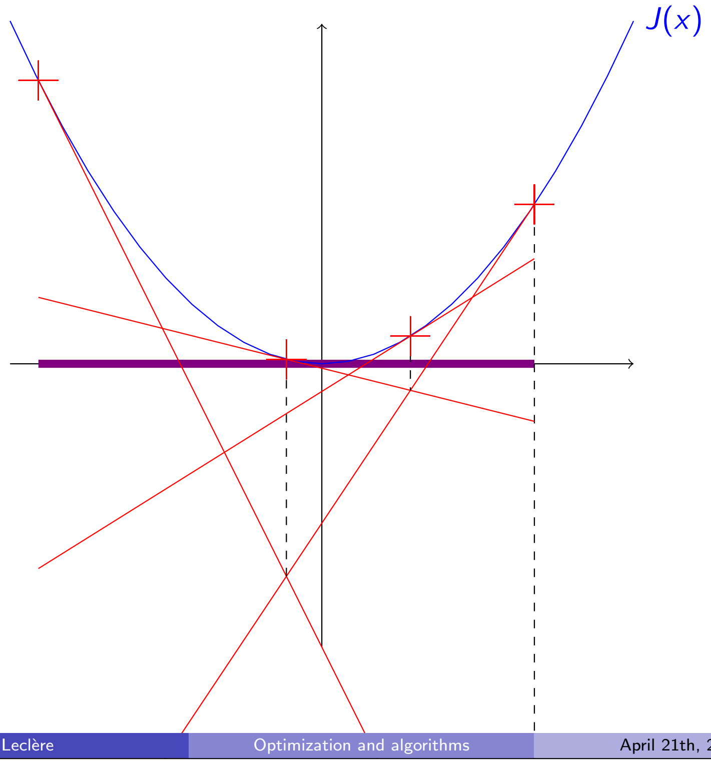
Another class of algorithm consists in constructing a simple model of the objective function f that is optimized and then refined.

Generally speaking, model-based algorithm goes as follows:

1. Solve $\min_{x\in X} f^k(x)$
2. Update model f^k into f^{k+1}

This approach might work if

- The model problem $\min_{x\in X} f^k(x)$ is *simple*
- The model f^k locally looks like the true function f around the optimum



Kelley algorithm

```
Data: Convex objective function $f$, Compact set $X$, Initial point
      $x_0 \in X$
Result: Admissible solution $x^{(k)}$, lower-bound $\underline{v}^{(k)}$
Set $f^{(0)} \equiv -\infty$ ;
for $k \in \mathbb{N}$ do
    Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$ ;
    Define a cut $\mathcal{C}^{(k)} : x \mapsto f(x^{(k)}) + \langle g^{(k)}, x - x^{(k)} \rangle$;
    Update the lower approximation $f^{(k+1)} = \max\{f^{(k)}, \mathcal{C}^{(k)}\}$ ;
    Solve $(P^{(k)}) : \quad \min_{x\in X} f^{(k+1)}(x)$;
    Set $\underline{v}^{(k)} = val(P^{(k)})$;
    Select $x^{(k+1)} \in sol(P^{(k)})$;
end
```

Algorithm 1: Kelley's cutting plane algorithm

Trust region method

Consider an unconstrained, non-linear, smooth problem

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \qquad f(x)$$

The idea of trust region is based on the following two facts:

- f locally looks like it's second order limited development

$$f(x+h) = f(x) + \langle \nabla f(x), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x) h + o(\|h\|^2)$$

- we know how to compute the minimum of a quadratic function on a ball.

Trust region method

The trust region method goes as follows, given a current point x^k and *trust radius* Δ^k

1. compute $f^k(x^k + h) = f(x^k) + \langle \nabla f(x^k), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x^k) h$
2. solve $\min_{y \in B(x^k, \Delta^k)} f^k(y)$, with optimal solution y^k
3. compute $f(y^k)$
4. compute the *concordance* r^k as the ratio actual decrease / model decrease
$$r^k = \frac{f(x^k) - f(y^k)}{f^k(x^k) - f^k(y^k)}$$
    - If r^k is small, the model is bad and you decrease Δ^k and restart the iteration
    - If r^k is large (close to 1) update the current point $x^{k+1} = y^k$.
5. If r^k is close to one and y^k is on the boundary, increase Δ^k.

$\rightsquigarrow$ there are full books on trust region methods.

What you have to know

- Important elements defining an optimization problem : continuous/discrete, smooth/non-differentiable, convex/non-convex, linear/non-linear, constrained/unconstrained.
- Main optimization classes: LP, MILP, differentiable unconstrained, combinatorial.
- The difference between heuristic and exact methods
- Main classes of exact method : descent direction, approximation method.

What you really should know

- Other important classes of optimization problem (LS, QP, SOCP, SDP)
- Some ideas of heuristic methods (simulated annealing, genetic algorithms)
- Kelley's cutting plane algorithm
- Principle of trust region method

What you have to be able to do

- Recognise a LP / MILP
- Recognise a (convex) differentiable optimization problem, constrained or not

What you should be able to do

- know how to use a "lift" variable, e.g.

$$
\underset{x}{\text{Min}} \quad \max(f_1(x), f_2(x)) = \quad \begin{aligned} \underset{x,z}{\text{Min}} \quad & z \\ \text{s.t.} \quad & f_1(x) \leq z \\ & f_2(x) \leq z \end{aligned}
$$

Gradient algorithms

V. Leclère (ENPC)

April 28th, 2023

Why should I bother to learn this stuff?

- Gradient algorithm is the easiest, most robust optimization algorithm. It is not numerically efficient, but numerous more advanced algorithm are built on it.
- Conjugate gradient algorithm(s) are efficient methods for (quasi)-quadratic function. They are in particular used for approximately solving large linear systems.
- $\Longrightarrow$ useful for comprehension of
  - more advanced continuous optimization algorithms
  - machine learning training methods
  - numerical methods for solving discretized PDE

Contents

1. Descent methods and black-box optimization [BV 9.1]
   - Some general thoughts and definition
   - Descent methods
2. Strong convexity consequences [BV 9.2]
3. Gradient descent [BV 9.3-9.4]
4. Conjugate gradient [JCG - 8.2]

Contents

1. Descent methods and black-box optimization [BV 9.1]
   - Some general thoughts and definition
   - Descent methods
2. Strong convexity consequences [BV 9.2]
3. Gradient descent [BV 9.3-9.4]
4. Conjugate gradient [JCG - 8.2]

A word on solution

- In this lecture, we are going to address unconstrained, finite dimensional, non-linear, smooth, optimization problem.
- In continuous non-linear (and non-quadratic) optimization, we cannot expect to obtain an *exact* solution. We are thus looking for approximate solutions.
- By solution, we generally mean local minimum.[1]
- The speed of convergence of an algorithm is thus determining an upper bound on the number of iterations required to get an ε-solution, for $\varepsilon > 0$.

[1] Sometimes just stationary points. Equivalent to global minimum in the convex setting.

Black-box optimization ♡

We consider the following unconstrained optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \qquad f(x)$$

- The black-box model consists in considering that we only know the function f through an oracle, that is a way of computing information on f at a given point x.
- Oracle gives local information on f. Oracles are generally given as user-defined code.
  - A *zeroth* order oracle only return the value $f(x)$.
  - A *first* order oracle return both $f(x)$ and $\nabla f(x)$.
  - A *second* order oracle return $f(x)$, $\nabla f(x)$ and $\nabla^2 f(x)$.
- By opposition, structured optimization leverage more knowledge on the objective function f. Classical models are
  - $f(x) = \sum_{i=1}^N f_i(x)$;
  - $f(x) = f_0(x) + \lambda g(x)$, where $f_0(x)$ is smooth and g is "simple", typically $g(x) = \|x\|_1$;
  - ...

Contents

1. Descent methods and black-box optimization [BV 9.1]
   - Some general thoughts and definition
   - Descent methods
2. Strong convexity consequences [BV 9.2]
3. Gradient descent [BV 9.3-9.4]
4. Conjugate gradient [JCG - 8.2]

Descent methods

Consider the unconstrained optimization problem

$$v^\sharp = \min_{x \in \mathbb{R}^n} \quad f(x).$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

For most of the analysis, we will assume f to be (strongly) convex, but the algorithms presented are often used in a non-convex setting.

To complete the algorithm, we need a stopping test, generally testing that $\|\nabla f(x^{(k)})\|$ is small enough.

Descent direction algorithms

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} < 0$, which can be seen from a first order development:

$$f(x^{(k)} + t^{(k)} d^{(k)}) = f(x^{(k)}) + t\big\langle \nabla f(x^{(k)}), d^{(k)} \big\rangle + o(t).$$

The most classical descent direction are[2]

1. $d^{(k)} = -\nabla f(x^{(k)})$ (gradient)
2. $d^{(k)} = -\nabla f(x^{(k)}) + \beta^{(k)} d^{(k-1)}$ (conjugate gradient)
3. $d^{(k)} = -\alpha^{(k)} \nabla f(x^{(k)}) + \beta^{(k)}(x^{(k)} - x^{(k-1)})$ (heavy ball ♢)
4. $d^{(k)} = -\big[\nabla^2 f(x^{(k)})\big]^{-1} \nabla f(x^{(k)})$ (Newton)
5. $d^{(k)} = -W^{(k)} \nabla f(x^{(k)})$ (Quasi-Newton)
   where $W^{(k)} \approx \big[\nabla^2 f(x^{(k)})\big]^{-1}$.

[2] they will be discussed at length during the course

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$,
  - too small and it will take forever
  - too large and it won't converge
- optimal $t^{(k)} \in \arg\min_{\tau \geq 0} f(x^{(k)} + \tau d^{(k)})$,
  - computing it requires solving an unidimensional problem
  - might not be worth the computation
- a backtracking or receeding step choice[3], for given $\tau_0 > 0, \alpha \in]0, 0.5[, \beta \in]0, 1[$,
  1. $\tau = \tau^0$
  2. if $f(x^{(k)} + \tau d^{(k)}) < f(x^{(k)}) + \alpha\tau \nabla f(x^{(k)})^\top d^{(k)}$: $t^{(k)} = \tau$, STOP
  3. $\tau \leftarrow \beta\tau$, go back to 2.
  - start with an "optimist" step τ_0
  - automatically adapts to ensure convergence
  - more complex procedure exists

[3] There exists a lot of other alternatives

Contents

1. Descent methods and black-box optimization [BV 9.1]
   - Some general thoughts and definition
   - Descent methods
2. Strong convexity consequences [BV 9.2]
3. Gradient descent [BV 9.3-9.4]
4. Conjugate gradient [JCG - 8.2]

Strong convexity definition(s)

Recall that $f : \mathbb{R}^n \to \mathbb{R}$ is m-convex[4] iff

$$f(tx+(1-t)y) \leq tf(x)+(1-t)f(y)-\frac{m}{2}t(1-t)\|y-x\|^2, \quad \forall x, y, \quad \forall t \in]0, 1[$$

If f is differentiable, it is m-convex iff

$$f(y) \geq f(x) + \big\langle \nabla f(x), y - x \big\rangle + \frac{m}{2}\|y - x\|^2, \qquad \forall y, x$$

If f is twice differentiable, it is m-convex iff

$$mI \preceq \nabla^2 f(x) \qquad \forall x$$

iff

$$m \leq \lambda \qquad \forall \lambda \in sp(\nabla^2 f(x)), \quad \forall x$$

$\rightsquigarrow$ this last characterization is the most usefull for our analysis.

[4] A strongly convex function is a m-convex function for some $m > 0$

Bounding the Hessian

Consider a m-convex $\mathcal{C}^2$ function (on its domain), and $x^{(0)} \in \mathrm{dom}\, f$. Denote $S := \mathrm{lev}_{f(x_0)}(f) = \big\{x \in \mathbb{R}^n \mid f(x) \le f(x_0)\big\}$.

As f is a strongly convex function S is bounded.

As $\nabla^2 f$ is continuous, there exists $M > 0$ such that, $\|\nabla^2 f(x)\| \le M$, for all $x \in S$.

Thus we have, for all $x \in S$,

$$mI \preceq \nabla^2 f(x) \preceq MI$$

Or equivalently

$$m \le \lambda_{min}(\nabla^2 f(x)) \le \lambda_{\max}(\nabla^2 f(x)) \le M \qquad \forall x \in S$$

Strongly convex suboptimality certificate

Let f be a m-convex $\mathcal{C}^2$ function. We have

$$f(y) \ge f(x) + \big\langle \nabla f(x) \,, y - x \big\rangle + \frac{m}{2}\|y - x\|^2, \qquad \forall y, x$$

The under approximation is minimized, for a given x, for $y^\sharp = x - \frac{1}{m}\nabla f(x)$, yielding

$$f(y) \ge f(x) - \frac{1}{2m}\|\nabla f(x)\|^2 \qquad \forall y$$

$$v^\sharp + \frac{1}{2m}\|\nabla f(x)\|^2 \ge f(x) \qquad \forall x$$

Thus we obtain the following sub-optimality certificate

$$\| \nabla f(x)\| \le \sqrt{2m\varepsilon} \quad \Longrightarrow \quad f(x) \le v^\sharp + \varepsilon$$

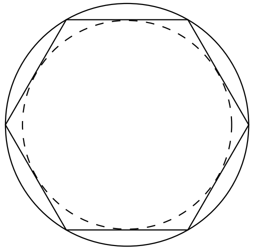
Condition numbers

For any $A \in S_n^{++}$ positive definite matrix, we define its condition number $\kappa(A) = \lambda_{max}/\lambda_{min} \ge 1$ the ratio between its largest and smallest eigenvalue.

Consider a bounded convex set C. Let D_{out} be the diameter of the smallest ball B_{out} containing C, and D_{in} be the diameter of the largest ball B_{in} contained in C.

Then the condition number of C is

$$\mathrm{cond}(C) = \Big(\frac{D_{out}}{D_{in}}\Big)^2$$



Condition number of sublevel set

We have, for all $x \in S$,

$$mI \preceq \nabla^2 f(x) \preceq MI$$

thus

$$\kappa(\nabla^2 f(x)) \le M/m$$

Further,

$$v^\sharp + \frac{m}{2}\|x - x^\sharp\|^2 \le f(x) \le v^\sharp + \frac{M}{2}\|x - x^\sharp\|^2$$

For any $v^\sharp \le \alpha \le f(x_0)$, we have

$$B\Big(x^\sharp, \sqrt{2(\alpha - v^\sharp)/M}\Big) \subset \mathrm{lev}_\alpha f \subset B\Big(x^\sharp, \sqrt{2(\alpha - v^\sharp)/m}\Big)$$

and thus

$$\mathrm{cond}(C_\alpha) \le M/m$$

Contents

1 Descent methods and black-box optimization [BV 9.1]
- Some general thoughts and definition
- Descent methods

2 Strong convexity consequences [BV 9.2]

3 Gradient descent [BV 9.3-9.4]

4 Conjugate gradient [JCG - 8.2]

Gradient descent ♡

- The gradient descent algorithm is a first-order descent direction algorithm with $d^{(k)} = -\nabla f(x^{(k)})$.
- That is, with an initial point x_0, we have

$$x^{(k+1)} = x^{(k)} - t^{(k)}\nabla f(x^{(k)}).$$

- The three step-size choices (fixed, optimal and decreasing) lead to variations of the algorithm.
- This algorithm is slow, but robust in the sense that it often ends up converging.
- Most implementations of advanced algorithms have fail-safe procedures that default to a gradient step when something goes wrong for numerical reasons.
- It is the basis of the stochastic-gradient algorithm, which is used (in advanced form) to train ML models.

Steepest descent algorithm ♢

- Using the linear approximation $f(x^{(k)} + h) = f(x^{(k)}) + \nabla f(x^{(k)})^\top h + o(\|h\|_{\maltese})$, it is quite natural to look for the steepest descent direction, that is

$$d^{(k)} \in \arg\min_{h} \quad \left\{ \nabla f(x^{(k)})^\top h \quad | \quad \|h\|_{\maltese} \leq 1 \right\}$$

- Here $\|\cdot\|_{\maltese}$ could be any norm on $\mathbb{R}^n$.
  - If $\|\cdot\|_{\maltese} = \|\cdot\|_2$, the steepest descent is a gradient step, i.e. proportional to $-\nabla f(x^{(k)})$.
  - If $\|\cdot\|_{\maltese} = \|\cdot\|_P$, $\|x\|_{\maltese} = \|P^{1/2}x\|_2$ for some $P \in S^n_{++}$, then the steepest descent is $-P^{-1}\nabla f(x^{(k)})$. In other words, a steepest descent step is a gradient step done on a problem after a change of variable $\bar{x} = P^{1/2}x$.
  - If $\|\cdot\|_{\maltese} = \|\cdot\|_1$, then the steepest descent can be chosen along a single coordinate, leading to the coordinate descent algorithm.

♠ Exercise: Prove these results.

Convergence results - convex case ♢

Assume that f is such that $0 \preceq \nabla^2 f \preceq MI$.

Theorem

The gradient algorithm with fixed step size $t^{(k)} = t \leq \frac{1}{M}$ satisfies

$$f(x^{(k)}) - v^\sharp \leq \frac{2M\|x^{(0)} - x^\sharp\|}{k} = O(1/k)$$

$\rightsquigarrow$ this is a *sublinear* rate of convergence.

Convergence results - strongly convex case ♢

Assume that f is such that $mI \preceq \nabla^2 f \preceq MI$, with $m > 0$. Define the conditioning factor $\kappa = M/m$.

Theorem

If $x^{(k)}$ is obtained from the optimal step, we have

$$f(x^{(k)}) - v^\sharp \leq C^k (f(x_0) - v^\sharp), \qquad C = 1 - 1/\kappa$$

If $x^{(k)}$ is obtained by receeding step size we have

$$f(x^{(k)}) - v^\sharp \leq C^k (f(x_0) - v^\sharp), \qquad C = 1 - \min\left\{2m\alpha, 2\beta\alpha\right\}/\kappa$$

$\rightsquigarrow$ linear rate of convergence.

Contents

1. Descent methods and black-box optimization [BV 9.1]
   - Some general thoughts and definition
   - Descent methods
2. Strong convexity consequences [BV 9.2]
3. Gradient descent [BV 9.3-9.4]
4. Conjugate gradient [JCG - 8.2]

Solving a linear system I

The gradient conjugate algorithm stems from looking for numerical solutions to the linear equation

$$Ax = b$$

- Never, ever, compute A^{-1} to solve a linear system.
- Classical algebraic method do a methodological factorization of A to obtain the (exact) value of x.
- These methods are in $O(n^3)$ operations. They only yield a solution at the end of the algorithm.
- The solution would be exact if there were no rounding errors...

Solving a linear system II

Alternatively, we can look to solve

$$\underset{x\in\mathbb{R}^n}{\mathrm{Min}} \qquad f(x) := \frac{1}{2} x^\top A x - b^\top x$$

which is a smooth, unconstrained, convex optimization problem, whose optimal solution is given by $Ax = b$.

We will assume that $A \in S^n_{++}$. If A is non symmetric, but invertible, we could consider $A^\top A x = A^\top b$.

Conjugate directions ◇

We say that $u, v \in \mathbb{R}^n$ are A-conjugate if they are orthogonal for the scalar product associated to A, i.e.

$$\langle u\,, v \rangle_A := u^\top A v = 0$$

Let $(\tilde{d}_i)_{i \in [k]}$ be a linearly independent family of vector. We can construct a family of conjugate directions $(d_i)_{i \in [k]}$ through the Gram-Schmidt procedure (without normalization), i.e., $\tilde{d}_1 = d_1$, and

$$d_\kappa = \tilde{d}_\kappa - \sum_{i=1}^{\kappa-1} \beta_{i,\kappa} d_i$$

where

$$\beta_{i,\kappa} = \frac{\langle \tilde{d}_\kappa\,, d_i \rangle_A}{\langle d_i\,, d_i \rangle_A} = \frac{\tilde{d}_\kappa^\top A d_i}{d_i^\top A d_i}$$

Conjugate direction method for quadratic function I ◇

Consider, for $A \in S^n_{++}$

$$f(x) := \frac{1}{2} x^\top A x - b^\top x$$

A conjugate direction algorithm is a descent direction algorithm such that,

$$x^{(k+1)} = \underset{x \in x_1 + E^{(k)}}{\arg\min} \quad f(x)$$

where

$$E^{(k)} = vect(d^{(1)}, \dots, d^{(k)})$$

♠ Exercise: Denote $g^{(k)} = \nabla f(x^{(k)})$. Show that

1. ${g^{(k)}}^\top d_i = 0$ for $i < k$
2. $g^{(k+1)} = g^{(k)} + t^{(k)} A d^{(k)}$
3. ${g^{(k)}}^\top d^{(i)} + t^{(k)} {d^{(k)}}^\top A d^{(i)} = 0$ for $i \leq k$
4. Either
   - ${g^{(k)}}^\top d^{(k)} = 0$ and $t^{(k)} = 0$
   - or ${g^{(k)}}^\top d^{(k)} < 0$ and $t^{(k)} = -\frac{{g^{(k)}}^\top d^{(k)}}{t^{(k)} {d^{(k)}}^\top A d^{(k)}}$

Conjugate direction method for quadratic function II ◇

```
Data: Linearly independent direction $\tilde{d}^{(1)}, \dots, \tilde{d}^{(n)}$, initial point $x^{(1)}$
Matrix $A$ and vector $b$
for $k \in [n]$ do
    $d^{(k)} = \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{\langle \tilde{d}^{(k)}, d^{(i)} \rangle_A}{\langle d^{(i)}, d^{(i)} \rangle_A} d^{(i)}$ ;          // A-orthogonalisation
    $t^{(k)} = \frac{\nabla f(x^{(k)})^\top d^{(k)}}{\langle d^{(k)}, d^{(k)} \rangle_A}$ ;          // optimal step
    $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$
```

Algorithm 1: Conjugate direction algorithm

This algorithm is such that (for a quadratic function f)

$$x^{(k+1)} = \underset{x \in x_1 + E^{(k)}}{\arg\min} \quad f(x)$$

where

$$E^{(k)} = vect(d^{(1)}, \dots, d^{(k)})$$

Conjugate gradient algorithm - quadratic function I ◇

The conjuate gradient algorithm set $\tilde{d}^{(k)} = -\underbrace{\nabla f(x^{(k)})}_{:= g^{(k)}}$.

In particular, we obtain that $E^{(k)} = vect(g^{(1)}, \dots, g^{(k)})$, and thus

$${g^{(k)}}^\top g^{(i)} = 0 \qquad \forall i \neq k$$

Note that

$$g^{(i+1)} - g^{(i)} = t^{(i)} A d^{(i)}, \quad \text{thus} \quad \frac{\langle \tilde{d}^{(k)}\,, d^{(i)} \rangle_A}{\langle d^{(i)}\,, d^{(i)} \rangle_A} = \frac{(\tilde{d}^{(k)})^\top (g^{(i+1)} - g^{(i)})}{{d^{(i)}}^\top (g^{(i+1)} - g^{(i)})}$$

Thus, through orthogonality we have

$$\begin{aligned} d^{(k)} &= \tilde{d}^{(k)} - \sum_{i=1}^{k-1} \frac{-{g^{(k)}}^\top (g^{(i+1)} - g^{(i)})}{{d^{(i)}}^\top (g^{(i+1)} - g^{(i)})} d^{(i)} \\ &= -g^{(k)} + \frac{{g^{(k)}}^\top (g^{(k)} - g^{(k-1)})}{{d^{(k-1)}}^\top (g^{(k)} - g^{(k-1)})} d^{(k-1)} = -g^{(k)} + \frac{\|g^{(k)}\|^2}{\|g^{(k-1)}\|^2} d^{(k-1)} \end{aligned}$$

Conjugate gradient algorithm - quadratic function II

```
Data: Initial point $x^{(1)}$, matrix $A$ and vector $b$
$g^{(1)} = Ax^{(1)} - b$ ;
$d^{(1)} = -g^{(1)}$ for $k = 2..n$ do
    If $\|g^{(k)}\|_2^2$ is small : STOP;
    $d^{(k)} = -g^{(k)} + \frac{\|g^{(k)}\|_2^2}{\|g^{(k-1)}\|_2^2} d^{(k-1)}$ ;
    $t^{(k)} = \frac{\|g^{(k)}\|_2^2}{{d^{(k)}}^\top A d^{(k)}}$ ;                    // optimal step
    $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$ ;
    $g^{(k+1)} = g^{(k)} + t^{(k)} A d^{(k)}$
```

Algorithm 2: Conjugate gradient algorithm - quadratic function

Conjugate gradient properties

We can show the following properties, for a quadratic function,

- The algorithm finds an optimal solution in at most n iterations
- If $\kappa = \lambda_{max}/\lambda_{min}$, we have

$$\|x^{(k+1)} - x^\sharp\|_A \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^k \|x^{(1)} - x^\sharp\|_A$$

- By comparison, gradient descent with optimal step yields

$$\|x^{(k+1)} - x^\sharp\|_A \leq 2\Big(\frac{\kappa-1}{\kappa+1}\Big)^k \|x^{(1)} - x^\sharp\|_A$$

Non-linear conjugate gradient

```
Data: Initial point $x^{(1)}$, first order oracle
for $k \in [n]$ do
    $g^{(k)} = \nabla f(x^{(k)})$ ;
    If $\|g^{(k)}\|_2^2$ is small : STOP;
    $d^{(k)} = -g^{(k)} + \beta^{(k)} d^{(k-1)}$ ;
    $t^{(k)}$ obtained by receeding linear search ;
    $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$ ;
```

Algorithm 3: Conjugate gradient algorithm - non-linear function

Two natural choices for the choice of β, equivalent for quadratic functions

- $\beta^{(k)} = \dfrac{\|g^{(k)}\|_2^2}{\|g^{(k-1)}\|_2^2}$ (Fletcher-Reeves)
- $\beta^{(k)} = \dfrac{{g^{(k)}}^\top (g^{(k)} - g^{(k-1)})}{\|g^{(k-1)}\|_2^2}$ (Polak-Ribière)

What you have to know

- What is a descent direction method.
- That there is a step-size choice to make.
- That there exists multiple descent direction.
- Gradient method is the slowest method, and in most case you should used more advanced method through adapted library.
- Conditionning of the problem is important for convergence speed.

What you really should know

- A problem can be pre-conditionned through change of variable to get faster results.
- Solving linear system can be done exactly through algebraic method, or approximately (or exactly) through minimization method.
- Conjugate gradient method are efficient tools for (approximately) solving a linear equation.
- Conjugate gradient works by exactly minimizing the quadratic function on an affine subspace.

What you have to be able to do

- Implement a gradient method with receeding step-size.

What you should be able to do

- Implement a conjugate gradient method.
- Use the strongly convex and/or Lipschitz gradient assumptions to derive bounds.

Newton and Quasi-Newton algorithms

V. Leclère (ENPC)

May 26th, 2023

Why should I bother to learn this stuff?

- Newton algorithm is, in theory, the best black-box algorithm for smooth strongly convex function. It is used in practice as well as a stepping step for more advanced algorithm.
- Quasi-Newton algorithms (in particular L-BFGS) are the actual by default algorithm for most smooth black-box optimization library. Used in large scale application (e.g. weather forecast) for decades.
- $\Longrightarrow$ useful for
  - understanding the optimization software you might use as an engineer
  - understanding more advanced methods (e.g. interior points methods)
  - getting an idea of why the convergence might behave strangely in practice

Oriented sum-up of previous courses

- There are two large classes of unconstrained, exact, black-box, optimization algorithms:
  - descent direction algorithm: $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
  - model based approach: $x^{(k+1)} = \arg\min_{x} f^{(k)}(x)$.
- We saw that defining a descent direction algorithm requires:
  - a direction $d^{(k)}$;
  - a step $t^{(k)}$;
  - a stopping test (e.g. $\|\nabla f(x^{(k)})\|_2 \ll 1$)
- We discussed gradient and conjugate gradient algorithms defined by $d^{(k)} = -\nabla f(x^{(k)}) + \beta^{(k)} d^{(k-1)}$:
  - convergence speed is sensitive to conditionning of the problem (i.e. if level sets are almost spherical);
  - you can precondition the problem through a change of coordinates;
  - can be interpreted as steepest descent method: $d^{(k)} = \arg\min_{\|d\|_P \leq 1} \nabla f(x^{(k)})^\top d$

Contents

1. Newton algorithm [BV 9.5]
   - Algorithm presentation, intuition and property
   - (Damped) Newton algorithm convergence
2. Quasi Newton [JCG - 11.2]
   - Quasi-Newton methods
   - BFGS algorithm

Contents

1. Newton algorithm [BV 9.5]
   - Algorithm presentation, intuition and property
   - (Damped) Newton algorithm convergence
2. Quasi Newton [JCG - 11.2]
   - Quasi-Newton methods
   - BFGS algorithm

Newton algorithm ♡

Let f be $\mathcal{C}^2$ such that $\nabla^2 f(x) \succ 0$ for all x (so in particular strictly convex).

The Newton algorithm is a descent direction algorithm with :

- $d^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)})$
- $t^{(k)} = 1$

Note that

$$\nabla f(x^{(k)})^\top d^{(k)} = -\nabla f(x^{(k)})^\top [\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)}) < 0$$

(unless $\nabla f(x^{(k)}) = 0$)

$\rightsquigarrow$ $d^{(k)}$ is a descent direction.

We are now going to give multiple justifications for this direction choice.

Second-order approximation minimization ♡

We have

$$f(x^{(k)} + d) = f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d + o(\|d\|^2)$$

The Newton method chooses the direction d (with step 1) that minimizes this second-order approximation, which is given by

$$\nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d^{(k)} = 0$$

$\rightsquigarrow$ The Newton method can be seen as a model-based method, where the model at iteration k is simply the second-order approximation.

$\rightsquigarrow$ A trust region method with confidence radius $+\infty$ is simply the Newton method.

Steepest descent with adaptative norm ◇

- The Newton direction $d^{(k)}$ is the steepest descent direction for the quadratic norm associated to $\nabla^2 f(x^{(k)})$:

$$d^{(k)} = \arg\min_d \left\{ \nabla f(x^{(k)})^\top d \quad | \quad \|d\|_{\nabla^2 f(x^{(k)})} \leq 1 \right\}$$

- Recall that the steepest gradient descent for a quadratic norm $\|\cdot\|_P$ converges rapidly if the condition number of the Hessian, after a change of coordinate, is small.
- In particular a good choice near $x^\sharp$ is $P = \nabla^2 f(x^\sharp)$.

$\rightsquigarrow$ fast around $x^\sharp$

Solution of linearized optimality condition ♢

The optimality condition is given by

$$\nabla f(x^\sharp) = 0$$

We can linearize it as

$$\nabla f(x^{(k)} + d) \approx \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d = 0$$

And the Newton step $d^{(k)}$ is the solution of this linearization.

Affine invariance ♢

- Recall that gradient and conjugate gradient methods can be accelerated through smart affine changes of variables (pre-conditioning).
- It is not the same for the Newton method:
  - Let A be an invertible matrix, and denote $y = Ax + b$, and $\tilde{f} : x \mapsto f(Ax + b)$.
  - $\nabla \tilde{f}(y) = A \nabla f(x)$ and $\nabla^2 \tilde{f}(y) = A^\top \nabla^2 f(x) A$
  - The Newton step for $\tilde{f}$ is thus

    $$d_y = -(A^\top \nabla^2 f(x) A)^{-1} A \nabla f(x) = -A^{-1}(\nabla^2 f(x))^{-1} \nabla f(x) = A^{-1} d_x$$

  - Consequently

    $$x^{(k+1)} - x^{(k)} = A(y^{(k+1)} - y^{(k)})$$

Contents

1. Newton algorithm [BV 9.5]
   - Algorithm presentation, intuition and property
   - (Damped) Newton algorithm convergence
2. Quasi Newton [JCG - 11.2]
   - Quasi-Newton methods
   - BFGS algorithm

Damped Newton algorithm ♡

```
Data: Initial point $x^{(0)}$, second-order oracle, error $\varepsilon > 0$.
while $\|\nabla f(x^{(k)})\| \geq \varepsilon$ do
    Solve for $d^{(k)}$
        $$\nabla^2 f(x^{(k)}) d^{(k)} = -\nabla f(x^{(k)})$$
    Compute $t^{(k)}$ by backtracking line-search, starting from $t = 1$;
    $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$
```

Algorithm 1: Damped Newton algorithm

- The Newton algorithm with fixed step size $t = 1$ is too numerically unstable, and you should always use a backtracking line-search.
- If the function is not strictly convex the Newton direction is not necessarily a descent direction, and you should check for it (and default to a gradient step).

Convergence idea ♢

Assume that f is strongly convex, such that $mI \preceq \nabla^2 f(x) \preceq MI$, and that the Hessian $\nabla^2 f$ is L-Lipschitz.
We can show that there exists $0 < \eta \leq m^2/L$ and $\gamma > 0$ such that

- If $\|\nabla f(x^{(k)})\|_2 \geq \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$$
- If $\|\nabla f(x^{(k)})\|_2 < \eta$, then $t^{(k)} = 1$ and
$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \Big(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\Big)^2$$

Newton is fast around the solution ♢

We have, if $\|\nabla f(x^{(k)})\|_2 < \eta$, then $t^{(k)} = 1$ and
$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \Big(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\Big)^2$$
Let $k = k_0 + \ell$, $\ell \geq 1$, with k_0 such that $\|\nabla f(x^{(k_0)})\|_2 < \eta$. Then $\|\nabla f(x^{(k)})\|_2 < \eta$, and,
$$\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2 \leq \Big(\frac{L}{2m^2}\|\nabla f(x^{(k-1)})\|_2\Big)^2$$
Recursively,
$$\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2 \leq \Big(\frac{L}{2m^2}\|\nabla f(x^{(k_0)})\|_2\Big)^{2^\ell} \leq \frac{1}{2^{2^\ell}}$$
And thus
$$f(x^{(k)}) - v^\sharp \leq \frac{1}{2m}\|\nabla f(x^{(k)})\|_2^2 \leq \frac{2m^3}{L^2}\frac{1}{2^{2^{\ell-1}}}$$
$\rightsquigarrow$ in the quadratic convergence phase, Newton's algorithm gets the result in a few iterations (5 or 6).

Convergence speed - Wrap-up

The Newton algorithm, for strongly convex function, have two phases :

- The damped phase, where $t^{(k)}$ can be less than 1. Each iteration yields an absolute improvement of $-\gamma < 0$.
- The quadratic phase, where each step $t^{(k)} = 1$.

Thus, the total number of iterations to get an ε solution is bounded above by
$$\frac{f(x^{(0)}) - v^\sharp}{\gamma} + \underbrace{\log_2(\log_2(\varepsilon_0/\varepsilon))}_{\lesssim 6}$$
where $\varepsilon_0 = 2m^3/L^2$.

Note that, in 6 iterations in the quadratic convergent phase we get an error $\varepsilon \approx 5.10^{-20}\varepsilon_0$.

Newton's properties in a nutshell ♡

- Full Newton step : $x^{(k+1)} = -[\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)})$
- Can be seen through various lenses:
  1. $[\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)})$ is a descent direction (f is strongly convex);
  2. model-based algorithm where the model is the second-order approximation;
  3. preconditioned gradient algorithm, with adaptive preconditioning.
- Is incredibly fast around the optimal solution.
- Far from the optimum a full Newton step is a bad idea:
  - If f is not strongly convex the Newton direction might not be a descent direction[1]!
  - $\rightsquigarrow$ check if it is a descent direction, otherwise make a gradient step.
  - Even with convexity the step might be too aggressive, $\rightsquigarrow$ receeding step choice.
- Convergence of the (damped) Newton's algorithm is in two phases:
  - slow constant update far from the optimum,
  - fast updates with full step close to the optimum.

[1] It can, for example, get you to the maximum of the second-order approximation...

Contents

1 Newton algorithm [BV 9.5]
- Algorithm presentation, intuition and property
- (Damped) Newton algorithm convergence

2 Quasi Newton [JCG - 11.2]
- Quasi-Newton methods
- BFGS algorithm

Contents

1 Newton algorithm [BV 9.5]
- Algorithm presentation, intuition and property
- (Damped) Newton algorithm convergence

2 Quasi Newton [JCG - 11.2]
- Quasi-Newton methods
- BFGS algorithm

The main idea ♡

Newton's step is very efficient (near optimality) but has three drawbacks:

1. having a second-order oracle to compute the Hessian
2. storing the Hessian (n^2 values)
3. solving a (dense) linear system : $\nabla^2 f(x^{(k)})d = -\nabla f(x^{(k)})$

The main idea of Quasi Newton method is to define $M^{(k)} \approx \nabla^2 f(x^{(k)})$ (or $W^{(k)} \approx [\nabla^2 f(x^{(k)})]^{-1}$):

1. from first order information $\rightsquigarrow$ no need to compute Hessian;
2. sparse $\rightsquigarrow$ smaller storage requirements;
3. $d^{(k)} = -W^{(k)}\nabla f(x^{(k)})$ $\rightsquigarrow$ no linear system solving.

Conditions on the approximate Hessian I ◊

We want to construct $M^{(k)}$ an approximation of $\nabla^2 f(x^{(k)})$, leading to a quadratic model of f at iteration k

$$f^{(k)}(x) := f(x^{(k)}) + \left\langle \nabla f(x^{(k)}) \, , x - x^{(k)} \right\rangle + \frac{1}{2}(x - x^{(k)})^\top M^{(k)}(x - x^{(k)})$$

We ask that the gradient of the model $f^{(k)}$ and the true function to match at the current and last iterates:

$$\begin{cases} \nabla f^{(k)}(x^{(k)}) = \nabla f(x^{(k)}) \\ \nabla f^{(k)}(x^{(k-1)}) = \nabla f(x^{(k-1)}) \end{cases}$$

This simply write as the Quasi-Newton equation

$$M^{(k)} \underbrace{(x^{(k)} - x^{(k-1)})}_{\delta_x^{(k-1)}} = \underbrace{\nabla f(x^{(k)}) - \nabla f(x^{(k-1)})}_{\delta_g^{(k-1)}}$$

♣ Exercise: prove it

Conditions on the approximate Hessian II ◇

We are looking for a matrix M such that

- $M \succ 0$
- $M\delta_x = \delta_g$ (only possible if $\delta_g^\top \delta_x > 0$ ♣ Exercise: prove it)
- $M^\top = M$
- M is constructed from first order information only
- If possible, M is sparse

$\rightsquigarrow$ an infinite number of solutions as we have $n(n+1)/2$ variables and n constraints.

$\rightsquigarrow$ Numerous quasi-Newton algorithms developed and tested between 1960-1980.

Choosing the approximate Hessian $M^{(k)}$ ◇

At the end of iteration k we have determined

- $x^{(k+1)}$ and $\delta_x^{(k)} = x^{(k+1)} - x^{(k)}$
- $g^{(k+1)} = \nabla f(x^{(k)})$ and $\delta_g^{(k)} = g^{(k+1)} - g^{(k)}$

and we are looking for $M^{(k+1)} \approx \nabla^2 f(x^{(k+1)})$ satisfying the previous requirement.

The idea is to choose $M^{(k+1)}$ close to $M^{(k)}$, that is to solve (analytically)

$$\begin{aligned} \underset{M \in S^n_{++}}{\text{Min}} \quad & d(M, M^{(k)}) \\ s.t. \quad & M\delta_x^{(k)} = \delta_g^{(k)} \end{aligned}$$

for some distance d.

Contents

1. Newton algorithm [BV 9.5]
   - Algorithm presentation, intuition and property
   - (Damped) Newton algorithm convergence
2. Quasi Newton [JCG - 11.2]
   - Quasi-Newton methods
   - BFGS algorithm

BFGS ◇

Broyden-Fletcher-Goldfarb-Shanno chose

$$d(A, B) := \mathrm{tr}(AB) - \ln\det(AB)$$

A few remarks

- $\Psi : M \mapsto \mathrm{tr}\, M - \ln\det(M)$ is convex on S^n_{++}
- For $M \in S^n_{++}$, $\mathrm{tr}\, M - \ln\det(M) = \sum_{i=1}^n \lambda_i - \ln(\lambda_i)$
- Ψ is minimized in the identity matrix
- $d(A, B) - n$ is the Kullback-Lieber divergence between $\mathcal{N}(0, A)$ and $\mathcal{N}(0, B)$

BFGS update

One of the pragmatic reasons for this choice of distance is that the optimal solution can be found analytically.

We have[2] (to alleviate notation we drop the index k on $\delta_x^{(k)}$ and $\delta_g^{(k)}$)

$$M^{(k+1)} = M^{(k)} + \frac{\delta_g {\delta_g}^\top}{{\delta_g}^\top \delta_x} - \frac{M^{(k)} \delta_x {\delta_x}^\top M^{(k)}}{{\delta_x}^\top M^{(k)} \delta_x}$$

Even better, denoting $W = M^{-1}$, we can show[3] that:

$$W^{(k+1)} = \Big(I - \frac{\delta_x {\delta_g}^\top}{{\delta_g}^\top \delta_x}\Big) W^{(k)} \Big(I - \frac{\delta_g {\delta_x}^\top}{{\delta_g}^\top \delta_x}\Big) + \frac{\delta_x {\delta_x}^\top}{{\delta_g}^\top \delta_x}$$

[2] with some effort

[3] fastidiously

BFGS algorithm

```
Data: Initial point $x^{(0)}$, First order oracle, error $\varepsilon > 0$.
$W^{(0)} = I$;
while $\|\nabla f(x^{(k)})\| \geq \varepsilon$ do
    $g^{(k)} := \nabla f(x^{(k)})$;
    $d^{(k)} := -W^{(k)} g^{(k)}$;
    Compute $t^{(k)}$ by backtracking line-search, starting from $t = 1$;
    $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
    $\delta_g = g^{(k+1)} - g^{(k)}$, $\delta_x = x^{(k+1)} - x^{(k)}$;
    $W^{(k+1)} = \Big(I - \frac{\delta_x \delta_g^\top}{\delta_g^\top \delta_x}\Big) W^{(k)} \Big(I - \frac{\delta_g \delta_x^\top}{\delta_g^\top \delta_x}\Big) + \frac{\delta_x \delta_x^\top}{\delta_g^\top \delta_x}$;
    $k = k + 1$;
```

Algorithm 2: BFGS algorithm

✔ First order oracle only

✔ No need to solve a linear system

✖ Still large memory requirement

✔ Convergence comparable to Newton's algorithm

Limited-memory BFGS (L-BFGS)

- For $n \geq 10^3$ storing the matrices is a difficulty.
- Instead of storing and updating the matrix $W^{(k)}$ we store (δ_x, δ_g) pairs.
- We can then compute $d^{(k)} = -W^{(k)} g^{(k)}$ directly from the last 5 to 20 pairs, using recursively the update rule and never computing $W^{(k)}$.

$\rightsquigarrow$ An algorithm with:

✔ First order oracle only

✔ No need to solve a linear system

✔ Same storage requirement as gradient algorithm

✔ Convergence comparable to Newton's algorithm

$\rightsquigarrow$ this is the "go to" algorithm when you want high-level precision for strongly convex smooth problems. It is the default choice in a lot of optimization libraries.

What you have to know

- At least one idea behind Newton's algorithm.
- The Newton step.
- That quasi-Newton methods are almost as good as Newton, without requiring a second order oracle.

What you really should know

- Newton's algorithm default step is 1, but you should use backtracking step anyway.
- Newton's algorithm converges in two phases : a slow damped phase, and a very fast quadratically convergent phase close to the optimum (at most 6 iterations).
- BFGS is the by default quasi-Newton method. It work by updating an approximation of the inverse of the Hessian close to the precedent approximation and satisfying some natural requirement.
- L-BFGS limit the memory requirement by never storing the matrix but only the step and gradient updates.

What you have to be able to do

- Implement a damped Newton method.

What you should be able to do

- Implement a BFGS method (with the update formula in front of your eyes)

Constrained optimization

V. Leclère (ENPC)

May 26th, 2023

Why should I bother to learn this stuff?

- Most real problems have constraints that you have to deal with.
- This course give a snapshot of the tools available to you.
- $\Longrightarrow$ useful for
  - having an idea of what can be done when you have constraints

Constrained optimization problem

- In the previous courses we have developed algorithms for unconstrained optimization problem.
- We now want to sketch some methods to deal with the constrained problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

- We are going to discuss multiple types of constraints set X:
  - X is a ball : $\{x \mid \|x - x_0\|_2 \leq r\}$
  - X is a box : $\{x \mid \underline{x}_i \leq x_i \leq \bar{x}_i \quad \forall i \in [n]\}$
  - X is a polyhedron: $\{x \mid Ax \leq b\}$
  - X is given through explicit constraints $\{x \mid g(x) = 0, \quad h(x) \leq 0\}$

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction
2. From constraints to cost
   - Penalization
   - Dualization

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction
2. From constraints to cost
   - Penalization
   - Dualization

Admissible descent direction

- Recall that a descent direction d at point $x^{(k)} \in \mathbb{R}^n$ is a vector such that $\nabla f(x^{(k)})^\top d < 0$.
- An admissible descent direction at point $x^{(k)} \in X$ is a descent direction $d \in \mathbb{R}^n$ such that,

$$\exists \varepsilon > 0, \quad \forall t \leq \varepsilon, \qquad x^{(k)} + td \in X.$$

- In other words, an admissible descent direction, is a direction that locally decreases the objective while staying in the constraint set.
- An admissible descent direction algorithm is naturally defined by:
  - A choice of admissible descent direction $d^{(k)}$
  - A choice of (sufficiently small) step $t^{(k)}$
  - $x^{(k+1)} = x^{(k)} + t^{(k)}d^{(k)} \in X$
- Warning: this does not necessarily converge. We can construct examples where the step size gets increasingly small because of the constraints.

A counter example

Consider

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & f(x) := \frac{4}{3}(x_1^2 - x_1x_2 + x_2^2)^{3/4} - x_3 \\ \text{s.t.} \quad & x \geq 0 \end{aligned}$$

We set $x^{(0)} = (0, 2^{-3/2}, 0)$, and $d^{(k)}$ such that $d_i^{(k)} = -g_i^{(k)}\mathbb{1}_{x_i^{(k)}>0}$, with $g_i^{(k)} = \nabla f(x^{(k)})$, and choose $t^{(k)}$ as the optimal step.

- This is an admissible direction descent with optimal step.
- f is strictly convex.
- $x^{(k)}$ converges toward a non-optimal point.

Conditional gradient algorithm

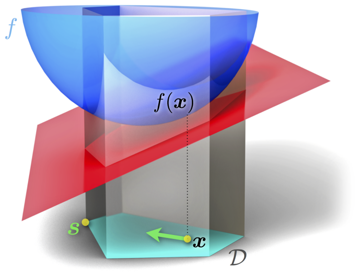
We address an optimization problem with a convex objective function f and compact polyhedral constraint set X, i.e.

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

where

$$X = \{x \in \mathbb{R}^n \quad | \quad Ax \leq b, \quad \tilde{A}x = \tilde{b}\}$$

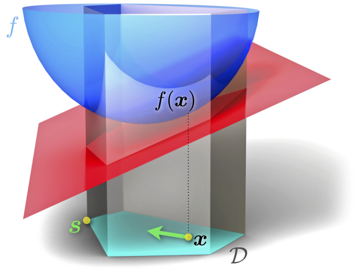
It is a descent algorithm, where we first look for an admissible descent direction $d^{(k)}$, and then look for the optimal step.



Conditional gradient algorithm

The conditional gradient method consists in choosing the descent direction that minimizes the linearization of f over X. More precisely, at step k we solve

$$y^{(k)} \in \arg\min_{y\in X} \quad f(x^{(k)})+\nabla f(x^{(k)})\cdot(y-x^{(k)}).$$



Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y\in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0,1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{y\in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction
2. From constraints to cost
   - Penalization
   - Dualization

Projection on a convex set ♡

Let $X \subset \mathbb{R}^n$ be a nonempty closed convex set. We call $P_X : \mathbb{R}^n \to \mathbb{R}^n$ the projection on X the fonction such that

$$P_X(x) = \arg\min_{x'\in X} \|x' - x\|_2^2$$

We have

- $\bar{x} = P_X(x)$ iff $(x - \bar{x}) \in N_X(\bar{x})$ (i.e. $\langle x - \bar{x} \,, x' - \bar{x}\rangle \leq 0, \quad \forall x' \in X$)
- $\langle P_X(y) - P_X(x) \,, y - x\rangle \geq 0$ (P_X is *non-decreasing*)
- $\|P_X(y) - P_X(x)\|_2 \leq \|y - x\|$ (P_X is a *contraction*)

♠ Exercise: Prove these results

Projected gradient I ♡

Consider

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Min}} \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

where f is differentiable and X convex.
The projected gradient algorithm generates the following sequence

$$x^{(k+1)} = P_X\left[x^{(k)} - t^{(k)} g^{(k)}\right]$$

Projected gradient II ♢

Theorem

Assume that $X \neq \emptyset$ is a closed convex set. $x^\sharp \in X$ is a critical point if and only if for one (or all) $t > 0$,

$$x^\sharp = P_X\left[x^\sharp - t\nabla f(x^\sharp)\right].$$

Theorem

If f is lower bounded on X, and with L-Lipschitz gradients, and X closed convex (nonempty) set. Then the projected gradient algorithm with step staying in $[a,b] \subset]0, 2/L[$, then $\|x^{(k+1)} - x^{(k)}\| \to 0$, and any adherence point of $\{x^{(k)}\}_{k\in\mathbb{N}}$ is a critical point.

Corollary: if f convex differentiable with L-Lipschitz gradient, X compact convex nonempty, the projected gradient algorithm with step $1/L$ is converging toward the optimal solution.

When to use ? ♡

- Projected gradient is useful only if the projection is simple, as projecting over a convex set consists in solving a constrained optimization problem.
- Projection is simple for balls and boxes.
- Finding an admissible direction is doable if the constraint set is polyhedral, or more generally conic-representable.

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction
2. From constraints to cost
   - Penalization
   - Dualization

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction

2. From constraints to cost
   - Penalization
   - Dualization

Idea of penalization ♡

We consider the constrained optimization problem

$$(\mathcal{P}) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x) \qquad \text{s.t.} \quad x \in X$$

and the following penalized version

$$(\mathcal{P}_r) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x) + r p(x)$$

Thus, a (constrained) problem is replaced by a sequence of (unconstrained) problems.

♣ Exercise: What is happening if $p = \mathbb{I}_X$?

Some monotonicity results ♢

$$(\mathcal{P}_r) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x) + r p(x)$$

The idea is that, with higher r, the penalization has more impact on the problem.

More precisely, let $0 < r_1 < r_2$, and x_{r_i} be an optimal solution of $(\mathcal{P}_{r_i})$.

We have:

- $p(x_{r_1}) \geq p(x_{r_2})$
- $f(x_{r_1}) \leq f(x_{r_2})$

♣ Exercise: prove these results.

Outer penalization

A first idea for choosing a penalization function p consists in choosing a function p such that:

- $p(x) = 0$ for $x \in X$
- $p(x) > 0$ for $x \notin X$

intuitively the idea is that p is the fine to pay for not respecting the constraint. Heuristically, it should be increasing with the distance to X.

Outer penalization - theoretical results ◇

Assume that

- p is l.s.c on $\mathbb{R}^n$
- $p \geq 0$
- $p(x) = 0$ iff $x \in X$

Further assume that f is l.s.c and there exists $r_0 > 0$ such that $x \mapsto f(x) + r_0 p(x)$ is coercive (i.e. $\to \infty$ if $\|x\| \to \infty$).
Then,

1. for $r > r_0$, $(\mathcal{P}_r)$ admit at least one optimal solution
2. $(x_r)_{r \to +\infty}$ is bounded
3. any adherence point of $(x_r)_{r \to +\infty}$ is an optimal solution of $\mathcal{P}$.

Outer penalization - quadratic case

Assume that

$$X = \{x \in \mathbb{R}^n \quad | \quad g(x) = 0, \quad h(x) \leq 0\}$$

then the quadratic penalization consists in choosing

$$p : x \mapsto \|g(x)\|^2 + \|(h(x))^+\|^2$$

This choice is interesting as (for affinely lower-bounded f):

- $x \mapsto f(x) + rp(x)$ is differentiable if f is differentiable
- $x_r \to x^\sharp$ if $r \to \infty$

However, generally speaking, if the constraints are impactful (e.g. have non-zero optimal multipliers), then

$$x_r \notin X$$

Outer penalization - L^1 case

Assume that

$$X = \{x \in \mathbb{R}^n \quad | \quad g(x) = 0, \quad h(x) \leq 0\}$$

another natural penalization consists in choosing

$$p : x \mapsto \|g(x)\|_1 + \|(h(x))^+\|_1$$

The interest of this approach is that, if the problem is convex and the constraints are qualified at optimality, then, for r large enough, an optimal solution to the penalized problem $(\mathcal{P}_r)$ is an optimal solution to the original problem $(\mathcal{P})$. Thus, we speak of exact penalization.

Unfortunately, this comes to the price of non-differentiability.

Inner penalization

Another approach consists in choosing a penalization function p that takes value $+\infty$ outside of X.

The idea here is to add a potential that repulses the optimal solution from the boundary.

This is typically done in a way to keep $f + \frac{1}{s}p$ smooth, and if possible convex.

Note that, for the inner penalization, we need the coefficient $\dfrac{1}{s} \to 0$, (hence $s \to +\infty$) for the penalized problem to converges toward the original one.

More on that in the next course.

Contents

1. Constructing an admissible trajectory
   - Admissible direction
   - Projected direction
2. From constraints to cost
   - Penalization
   - Dualization

Duality, here we go again ♡

Recall that to a primal problem

$$(\mathcal{P}) \quad \underset{x\in\mathbb{R}^n}{\text{Min}} \quad f(x) \tag{1}$$

$$\text{s.t.} \quad g(x) = 0 \tag{2}$$

$$h(x) \leq 0 \tag{3}$$

we associate the dual problem

$$(\mathcal{D}) \quad \underset{\lambda,\mu\geq 0}{\text{Max}} \quad \underbrace{\underset{x}{\text{Min}} \quad f(x) + \lambda^\top g(x) + \mu^\top h(x)}_{\Phi(\lambda,\mu)}$$

♣ Exercise: Under which sufficient conditions are these problems equivalent ?

Duality seen as exact penalization ♡

If $(\mathcal{P})$ is convex differentiable and the constraints are qualified, then for any optimal multiplier $\overline{\lambda}, \overline{\mu}$ the unconstrained problem

$$\underset{x}{\text{Min}} \qquad f(x) + \overline{\lambda}^\top g(x) + \overline{\mu}^\top h(x)$$

have the same optimal solution as the original problem $(\mathcal{P})$.

Projected gradient in the dual

Consider the dual problem

$$(\mathcal{D}) \quad \underset{\lambda,\mu\geq 0}{\text{Max}} \qquad \Phi(\lambda,\mu)$$

Recall that, under technical conditions,

$$\nabla\Phi(\lambda,\mu) = \begin{pmatrix} g(x^\sharp(\lambda,\mu)) \\ h(x^\sharp(\lambda,\mu)) \end{pmatrix}$$

where $x^\sharp(\lambda,\mu)$ is an optimal solution of the inner minimization problem for given λ, μ.

We suggest solving this problem through projected gradient with step t:

$$\lambda^{(k+1)} = \lambda^{(k)} + t g(x^\sharp(\lambda^{(k)}, \mu^{(k)}))$$

$$\mu^{(k+1)} = [\mu^{(k)} + t h(x^\sharp(\lambda^{(k)}, \mu^{(k)}))]^+$$

Uzawa's algorithm

```
Data: Initial primal point $x^{(0)}$, Initial dual points $\lambda^{(0)}, \mu^{(0)}$, unconstrained
      optimization method, dual step $t > 0$.
while $\|g(x^{(k)})\|_2 + \|(h(x^{(k)}))^+\|_2 \geq \varepsilon$ do
    Solve for $x^{(k+1)}$
        $\underset{x}{\text{Min}} \quad f(x) + \lambda^{(k)\top} g(x) + \mu^{(k)\top} h(x)$
    Update the multipliers
        $\lambda^{(k+1)} = \lambda^{(k)} + t g(x^{(k+1)})$
        $\mu^{(k+1)} = [\mu^{(k)} + t h(x^{(k+1)})]^+$
```

Algorithm 1: Uzawa algorithm

Convergence requires strong convexity and constraint qualifications.

Exercise : decomposition by prices

We consider the following energy problem:

- you are an energy producer with N production unit
- you have to satisfy a given demand planning for the next 24h (i.e. the total output at time t should be equal to d_t)
- the time step is the hour, and each unit has a production cost for each planning given as a convex quadratic function of the planning

1. Model this problem as an optimization problem. In which class does it belong? How many variables?
2. Apply Uzawa's algorithm to this problem. Why could this be an interesting idea?
3. Give an economic interpretation of this method.
4. What would happen if each unit had production constraints?

What you have to know

- There is three main ways of dealing with constraints:
  - choosing an admissible direction
  - projection of the next iterate
  - penalizing the constraints

What you really should know

- admissible direction methods are mainly usefull for polyhedral constraint set
- projection is usefull only if the admissible set is simple (ball or bound constraints)
- penalization can be inner or outer, differentiable or not.

What you have to be able to do

- Implement a penalization approach.

What you should be able to do

- Implement Uzawa's algorithm.

Interior Points Methods

V. Leclère (ENPC)

June 2nd, 2023

Why should I bother to learn this stuff?

- Interior point methods are competitive with simplex method for linear programm
- Interior point methods are state of the art for most conic (convex) problems
- $\Longrightarrow$ useful for
  - understanding what is used in numerical solvers
  - specialization in optimization

Contents

1. Recalls on convex differentiable optimization problems
2. Equality constrained optimization
3. Barrier methods [BV 11.2-11.3]
   - Interior penalization
   - Duality
   - Interpretation through KKT condition
4. Interior Point Method
5. Application to linear problem
6. Wrap-up

Convex differentiable optimization problem

We consider the following convex optimization problem

$$
\begin{aligned}
(\mathcal{P}) \qquad \min_{x\in\mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & Ax = b \\
& g_i(x) \leq 0 \qquad\qquad \forall i \in [\![1, n_I]\!]
\end{aligned}
$$

where A is a $n_E \times n$ matrix, and all functions f and g_i are assumed convex, real valued and twice differentiable.

Introducing the Lagrangian I ♡

$$
\begin{aligned}
(\mathcal{P}) \quad \min_{x\in\mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & Ax = b \\
& g_i(x) \leq 0 \qquad\qquad \forall i \in [\![1, n_I]\!]
\end{aligned}
$$

is equivalent to

$$\min_{x\in\mathbb{R}^n} \quad f(x) + \mathbb{I}_{\{Ax-b=0\}} + \sum_{i=1}^{n_I} \mathbb{I}_{\{h_i(x)\leq 0\}}$$

which we rewrite

$$\min_{x\in\mathbb{R}^n} \quad \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \quad f(x) + \sup_{\lambda\in\mathbb{R}^{n_E}} \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \sup_{\mu_i\geq 0} \mu_i h_i(x)$$

Introducing the Lagrangian II ♡

$$(\mathcal{P}) \quad \min_{x\in\mathbb{R}^n} \quad \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \quad \underbrace{f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i g_i(x)}_{:=\mathcal{L}(x;\lambda,\mu)}$$

$$(\mathcal{D}) \quad \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} \quad \min_{x\in\mathbb{R}^n} \quad f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i g_i(x)$$

As for any function ϕ we always have

$$\sup_y \inf_x \phi(x,y) \leq \inf_x \sup_y \phi(x,y)$$

we have that (weak duality)

$$val(\mathcal{D}) \leq val(\mathcal{P}).$$

Lower bounds from duality ♡

Define the dual function

$$d(\lambda,\mu) := \inf_x \mathcal{L}(x;\lambda,\mu)$$

Then we have $val(\mathcal{D}) = \sup_{\lambda\in\mathbb{R}^{n_E},\mu\in\mathbb{R}^{n_I}_+} d(\lambda,\mu)$.

Thus, we can compute a lower bound to $val(\mathcal{D}) \leq val(\mathcal{P})$ by choosing an any admissible dual points $\lambda \in \mathbb{R}^{n_E}, \mu \in \mathbb{R}^{n_I}_+$ and solving the unconstrained problem

$$d(\lambda,\mu) = \inf_{x\in\mathbb{R}^n} f(x) + \lambda^\top(Ax-b) + \sum_{i=1}^{n_I} \mu_i h_i(x)$$

Constraint qualification

Recall that, for a convex differentiable optimization problem, the constraints are qualified if *Slater's condition* is satisfied :

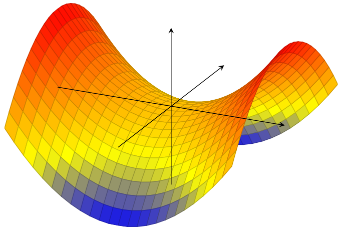
$$\exists x_0 \in \mathbb{R}^n, \qquad Ax_0 = b, \quad \forall i \in [n_I], \quad g_i(x_0) < 0$$

i.e.,, there exists a strictly admissible feasible point

Saddle point

If $(\mathcal{P})$ is a convex optimization problem with qualified constraints, then

- $val(\mathcal{D}) = val(\mathcal{P})$
- any optimal solution $x^\sharp$ of $(\mathcal{P})$ is part of a saddle point $(x^\sharp; \lambda^\sharp, \mu^\sharp)$ of $\mathcal{L}$
- $(\lambda^\sharp, \mu^\sharp)$ is an optimal solution of $(\mathcal{D})$



Karush Kuhn Tucker conditions

If Slater's condition is satisfied, then $x^\sharp$ is an optimal solution to (P) if and only if there exists optimal multipliers $\lambda^\sharp \in \mathbb{R}^{n_E}$ and $\mu^\sharp \in \mathbb{R}^{n_I}$ satisfying

$$\begin{cases} \nabla f(x^\sharp) + A^\top \lambda^\sharp + \sum_{i=1}^{n_I} \mu_i^\sharp \nabla g_i(x^\sharp) = 0 & \text{first order condition} \\ Ax^\sharp = b & \text{primal admissibility} \\ g(x^\sharp) \leq 0 & \\ \mu^\sharp \geq 0 & \text{dual admissibility} \\ \mu_i^\sharp g_i(x^\sharp) = 0, \quad \forall i \in [\![1, n_I]\!] & \text{complementarity} \end{cases}$$

The three last conditions are sometimes compactly written

$$0 \geq g(x^\sharp) \perp \mu \geq 0$$

Contents

1 Recalls on convex differentiable optimization problems

2 Equality constrained optimization

3 Barrier methods [BV 11.2-11.3]
- Interior penalization
- Duality
- Interpretation through KKT condition

4 Interior Point Method

5 Application to linear problem

6 Wrap-up

Intuition for Newton's method: unconstrained case

Newton's method is an iterative optimization method that minimizes a quadratic approximation of the objective function at the current point $x^{(k)}$. Consider the following unconstrained optimization problem:

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

At $x^{(k)}$ we have

$$f(x^{(k)} + d) = f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d + o(\|d\|^2)$$

And the direction $d^{(k)}$ minimizing the quadratic approximation is given by solving for d

$$\nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d = 0.$$

Intuition for Newton's method: constrained case ♡

Approximate the linearly constrained optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

by

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & f(x^{(k)}) + \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & A(x^{(k)} + d) = b \end{aligned}$$

Which is equivalent to solving (for given admissible $x^{(k)}$)

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & Ad = 0 \end{aligned}$$

Finding Newton's direction

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \nabla f(x^{(k)})^\top d + \frac{1}{2} d^\top \nabla^2 f(x^{(k)}) d \\ \text{s.t.} \quad & Ad = 0 \end{aligned}$$

By KKT the optimal $d^{(k)}$ is given by solving for (d, λ)

$$\begin{cases} \nabla f(x^{(k)}) + \nabla^2 f(x^{(k)}) d + A^\top \lambda = 0 \\ Ad = 0 \end{cases}$$

Or in a matricial form

$$\begin{pmatrix} \nabla^2 f(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x^{(k)}) \\ 0 \end{pmatrix}$$

Newton's algorithm: equality constrained case

```
Data: Initial admissible point $x_0$
Result: quasi-optimal point
$k = 0$;
while $|\nabla f(x^{(k)})| \geq \varepsilon$ do
    Solve for $d$
        $\begin{pmatrix} \nabla^2 f(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x^{(k)}) \\ 0 \end{pmatrix}$
    Line-search for $\alpha \in [0,1]$ on $f(x^{(k)} + \alpha d^{(k)})$
    $x^{(k+1)} = x^{(k)} + \alpha d^{(k)}$
    $k \leftarrow k + 1$
```

Algorithm 1: Newton's algorithm

Contents

1 Recalls on convex differentiable optimization problems

2 Equality constrained optimization

3 Barrier methods [BV 11.2-11.3]

- Interior penalization
- Duality
- Interpretation through KKT condition

4 Interior Point Method

5 Application to linear problem

6 Wrap-up

Contents

1 Recalls on convex differentiable optimization problems

2 Equality constrained optimization

3 Barrier methods [BV 11.2-11.3]
- Interior penalization
- Duality
- Interpretation through KKT condition

4 Interior Point Method

5 Application to linear problem

6 Wrap-up

Constrained optimization problem

We now want to consider a convex differentiable optimization problem with equality and inequality constraints.

$$(\mathcal{P}_\infty) \qquad \min_{x\in\mathbb{R}^n} \quad f(x)$$
$$\text{s.t.} \quad Ax = b$$
$$g_i(x) \leq 0 \qquad \forall i \in [\![1, n_I]\!]$$

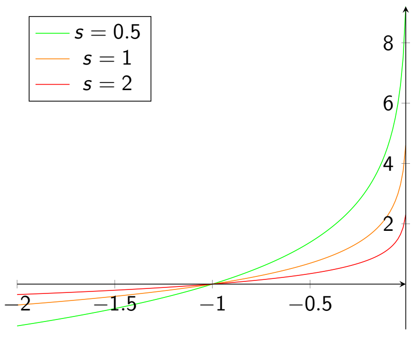
where all functions f and g_i are assumed convex, finite valued and twice differentiable.
Which we rewrite

$$\min_{x\in\mathbb{R}^n} \quad f(x) + \sum_{i=1}^{n_I} \mathbb{I}_{\mathbb{R}^-}(g_i(x))$$
$$\text{s.t.} \quad Ax = b$$

The negative log function

- The idea of barrier method is to replace the indicator function $\mathbb{I}_{\mathbb{R}^-}$ by a smooth function.
- We choose the function $z \mapsto -1/s\log(-z)$
- Note that they also take value $+\infty$ on $\mathbb{R}^+$

Illustration of barrier functions



Calculus

- We define
$$\phi : x \mapsto -\sum_{i=1}^{n_I} \ln(-g_i(x))$$
- Thus we have $\frac{1}{s}\phi(x) \xrightarrow[s\to+\infty]{} \mathbb{I}_{\{g_i(x)<0,\ \forall i\in[n_I]\}}$
- We have
$$\nabla\phi(x) = \sum_{i=1}^{n_I} -\frac{1}{g_i(x)}\nabla g_i(x)$$
$$\nabla^2\phi(x) = \sum_{i=1}^{n_I} \left[\frac{1}{g_i^2(x)}\nabla g_i(x)\nabla g_i(x)^\top - \frac{1}{g_i(x)}\nabla^2 g_i(x)\right]$$

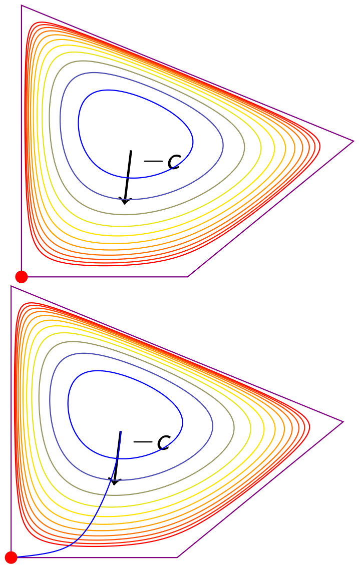
Penalized problem ♡

We consider

$$(\mathcal{P}_{\infty s}) \quad \min_{x\in\mathbb{R}^n} s f(x) + \frac{1}{s}\phi(x)$$
$$\text{s.t.} \quad Ax = b$$

with optimal solution $x^\sharp_s$.

Letting s goes to $+\infty$ get to solution of $(\mathcal{P})$ along the central path.



Characterizing central path I ♢

x_s is solution of

$$(\mathcal{P}_s) \quad \min_{x\in\mathbb{R}^n} s f(x) + \phi(x)$$
$$\text{s.t.} \quad Ax = b$$

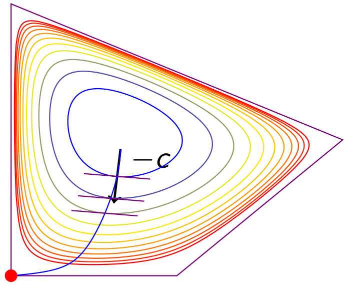
if and only if, there exists $\lambda_s \in \mathbb{R}^{n_E}$, such that

$$\begin{cases} Ax_s = b & \\ g_i(x_s) < 0 & \forall i \in [n_I] \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top\lambda = 0 & \end{cases}$$

Characterizing central path II ♢

$$\begin{cases} Ax_s = b \\ g(x_s) < 0 \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top\lambda = 0 \end{cases}$$

If $A = 0$ it means that $\nabla f(x_s)$ is orthogonal to the level lines of ϕ



Contents

1. Recalls on convex differentiable optimization problems
2. Equality constrained optimization
3. Barrier methods [BV 11.2-11.3]
   - Interior penalization
   - Duality
   - Interpretation through KKT condition
4. Interior Point Method
5. Application to linear problem
6. Wrap-up

Duality

Recall the original optimization problem

$$
\begin{aligned}
(\mathcal{P}_\infty) \qquad & \min_{x\in\mathbb{R}^n} \quad f(x) \\
& \text{s.t.} \quad Ax = b \\
& \qquad\;\; g_i(x) \leq 0 \qquad\qquad \forall i \in [\![1, n_I]\!]
\end{aligned}
$$

with Lagrangian

$$\mathcal{L}(x; \lambda, \mu) := f(x) + \lambda^\top (Ax - b) + \sum_{i=1}^{n_I} \mu_i g_i(x)$$

and dual function

$$d(\lambda, \mu) := \inf_{x\in\mathbb{R}^n} \mathcal{L}(x; \lambda, \mu).$$

For any admissible dual point $(\lambda, \mu) \in \mathbb{R}^{n_E} \times \mathbb{R}^{n_I}_+$, we have

$$d(\lambda, \mu) \leq val(\mathcal{P}_\infty)$$

Getting a lower bound

For given admissible dual point $(\lambda, \mu) \in \mathbb{R}^{n_E} \times \mathbb{R}^{n_I}_+$, a point $x^\sharp(\lambda, \mu)$ minimizing $\mathcal{L}(\cdot, \lambda, \mu)$, is characterized by first order conditions

$$\nabla f(x^\sharp(\lambda, \mu)) + A^\top \lambda + \sum_{i=1}^{n_I} \mu_i \nabla g_i(x^\sharp(\lambda, \mu)) = 0$$

which gives

$$d(\lambda, \mu) = \mathcal{L}(x^\sharp(\lambda, \mu); \lambda, \mu) \leq val(\mathcal{P}_\infty)$$

Dual point on the central path

Now recall that x_s, solution of $(\mathcal{P}_s)$, is characterized by

$$\begin{cases} Ax_s = b, g(x_s) < 0 \\ s\nabla f(x_s) + \nabla\phi(x_s) + A^\top \lambda_s = 0 \end{cases}$$

And we have seen that

$$\nabla\phi(x) = \sum_{i=1}^{n_I} \frac{1}{-g_i(x)} \nabla g_i(x)$$

Thus,

$$\nabla f(x_s) + A^\top \lambda_s / s + \sum_{i=1}^{n_I} \underbrace{\frac{1}{-s g_i(x_s)}}_{(\mu_s)_i} \nabla g_i(x_s) = 0$$

which means that $x_s = x^\sharp(\lambda_s / s, \mu_s)$.

Bounding the error

Let x_s be a primal point on the central path satisfying

$$\exists \lambda_s \in \mathbb{R}^{n_E}, \qquad s\nabla f(x_s) + \nabla\phi(x_s) + A^\top \lambda_s = 0$$

We define a dual point $(\mu_s)_i = \frac{1}{-s g_i(x_s)} > 0$. We have

$$
\begin{aligned}
d(\mu_s, \lambda_s/s) &= \mathcal{L}(x_s, \mu_s, \lambda_s/s) \\
&= f(x_s) + \frac{1}{s}\lambda_s{}^\top \underbrace{(Ax_s - b)}_{=0} + \sum_{i=1}^{n_I} \frac{1}{-s g_i(x_s)} g_i(x_s) \\
&= f(x_s) - \frac{n_I}{s} \leq val(\mathcal{P}_\infty)
\end{aligned}
$$

➥ x_s is an n_I/s-optimal solution of $(\mathcal{P}_\infty)$.

Contents

1. Recalls on convex differentiable optimization problems
2. Equality constrained optimization
3. Barrier methods [BV 11.2-11.3]
   - Interior penalization
   - Duality
   - Interpretation through KKT condition
4. Interior Point Method
5. Application to linear problem
6. Wrap-up

Interpretation through KKT condition

A point x_s is on the central path iff it is strictly admissible and there exists $\lambda \in \mathbb{R}^{n_E}$ such that

$$\nabla f(x_s) + A^\top \lambda + \sum_{i=1}^{n_I} \underbrace{\frac{1}{-s g_i(x)}}_{(\mu_s)_i} \nabla g_i(x) = 0$$

which can be rewritten

$$\begin{cases} \nabla f(x) + A^\top \lambda + \sum_{i=1}^{n_i} \mu_i \nabla g_i(x) = 0 \\ Ax = b, g_i(x) \leq 0 \\ \mu \geq 0 \\ -\mu_i g_i(x) = \frac{1}{s} & \forall i \in [n_I] \end{cases}$$

Contents

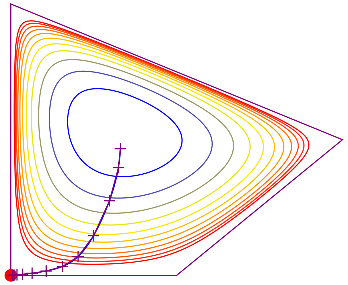
1. Recalls on convex differentiable optimization problems
2. Equality constrained optimization
3. Barrier methods [BV 11.2-11.3]
   - Interior penalization
   - Duality
   - Interpretation through KKT condition
4. Interior Point Method
5. Application to linear problem
6. Wrap-up

Taking a step back

- We saw that we can extend Newton's method to solve linearly constrained optimization problem.
- We saw that we can approximate inequality constraints through the use of logarithmic barrier $-1/s \sum_i \ln(-g_i(x))$.
- We proved that x_s is an n_I/s-optimal solution.
- The trade-off with s is : larger s means x_s closer to optimal solution x_∞ but the approximate problem $(\mathcal{P}_s)$ have worse conditionning.

Barrier method

```
Data: increase $\rho > 1$, error $\varepsilon > 0$, initial $t$
Result: $\varepsilon$-optimal point
solve $(\mathcal{P}_s)$ and set $x = x_s$ ;
while $n_I/t \geq \varepsilon$ do
    increase t: $t = \rho t$
    centering step: solve $(\mathcal{P}_s)$ starting at $x$ ;
    update : $x = x_s$
```



Question : why solve $(\mathcal{P}_s)$ to optimality ?

Solving $(\mathcal{P}_s)$ with Newton's method

$$(\mathcal{P}_s) \quad \min_{x\in\mathbb{R}^n} \quad s f(x) + \phi(x) \qquad \text{s.t.} \quad Ax = b$$

is a linearly constrained optimization problem that can be solved by Newton's method.
More precisely we have $x^{(k+1)} = x^{(k)} + d^{(k)}$ with $d^{(k)}$ a solution of

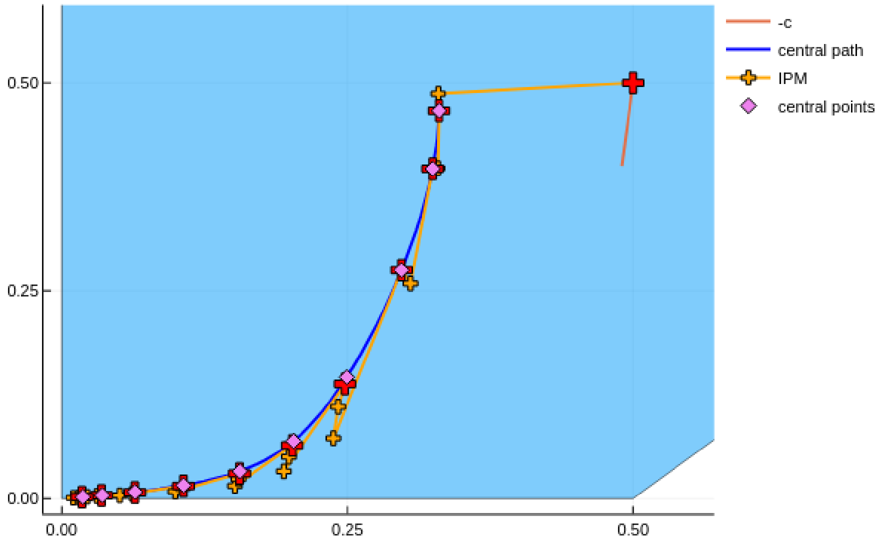
$$\begin{pmatrix} s\nabla^2 f(x^{(k)}) + \nabla^2\phi(x^{(k)}) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -s\nabla f(x^{(k)}) - \nabla\phi(x^{(k)}) \\ 0 \end{pmatrix}$$

Path following interior point method

```
Data: increase $\rho > 1$, error $\varepsilon > 0$, initial $s_0$
initial strictly feasible point $x_0$
$k = 0$
$x \leftarrow x_0$ , $s \leftarrow s_0$
for $k \in \mathbb{N}$ do                                  // Outer step
    for $\kappa \in [K]$ do                                 // Inner step
        solve for $d$ ;                            // Newton step for $(\mathcal{P}_s)$
            $\begin{pmatrix} s_k\nabla^2 f(x) + \nabla^2\phi(x) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = \begin{pmatrix} -s_k\nabla f(x) - \nabla\phi(x) \\ 0 \end{pmatrix}$
            reduce $\alpha$ from 1 until $f(x+\alpha d) \leq f(x)$;
        $x \leftarrow x + \alpha d$;
    $s \leftarrow \rho s$;
```

Algorithm 2: Path following algorithm

Path following algorithm



Contents

1. Recalls on convex differentiable optimization problems
2. Equality constrained optimization
3. Barrier methods [BV 11.2-11.3]
   - Interior penalization
   - Duality
   - Interpretation through KKT condition
4. Interior Point Method
5. Application to linear problem
6. Wrap-up

A linear problem - inequality form

We consider the following LP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & a_i^\top x \leq b_i \qquad\qquad \forall i \in [n_I] \end{aligned}$$

Where $a_i^\top = A[:, i]$ is the row of matrix A, such that the constraints can be written $Ax \leq b$.
Thus, x_s is the solution of

$$\min_{x\in\mathbb{R}^n} \quad sc^\top x + \phi(x)$$

where

$$\phi(x) := -\sum_{i=1}^{n_I} \ln(b_i - a_i^\top x)$$

Calculus ♢

$$\phi(x) = -\sum_{i=1}^{n_I} \ln(b_i - a_i^\top x)$$

$$\nabla\phi(x) = \sum_{i=1}^{n_I} \frac{1}{b_i - a_i^\top x} a_i$$

$$\nabla^2\phi(x) = \frac{1}{(b_i - a_i^\top x)^2} a_i a_i^\top$$

This can be written in matrix form, using the vector $d \in \mathbb{R}^{n_I}$ defined by $d_i = \frac{1}{b_i - a_i^\top x}$

$$\nabla\phi(x) = A^\top d$$

$$\nabla^2\phi(x) = A^\top diag(d)^2 A$$

Newton step ♢

Starting from x, the Newton direction for $(\mathcal{P}_s)$ is

$$dir_s(x) = -(\nabla^2\phi(x))^{-1}(sc + \nabla\phi(x))$$

which, in algebraic form, yields

$$dir_s(x) = -[A^\top diag(d)^2 A]^{-1}(sc + A^\top d)$$

with $d_i = 1/(b_i - a_i^\top x)$.

Theory tell us to use a step-size of 1 for Newton's method.

Practice teach us to use a smaller step-size (or linear-search).

Interior Point Method for LP pseudo code

```
Data: Initial admissible point $x_0$, initial penalization $s_0 > 0$;
parameter: $\rho > 1$, $N_{in} \geq 1$, $N_{out} \geq 1$;
Result: quasi-optimal point
$x = x0$, $s = s_0$;
for $k = 1..N_{out}$ do
    for $\kappa = 1..N_{in}$ do
        Compute $d$, with $d_i = 1/(b_i - a_i^T x)$;
        Solve for dir
            $$A^\top \mathrm{diag}(d)^2 A\mathrm{dir} = -(sc + A^\top d)$$
        reduce $\alpha$ from 1 until[a] $f(x + \alpha\mathrm{dir}) \leq f(x)$;
        update $x \leftarrow x + \alpha\mathrm{dir}$ ;
    update $s \leftarrow \rho s$;
```

Algorithm 3: Interior Point Method for LP

[a] simplest condition described here

Contents

1 Recalls on convex differentiable optimization problems

2 Equality constrained optimization

3 Barrier methods [BV 11.2-11.3]
- Interior penalization
- Duality
- Interpretation through KKT condition

4 Interior Point Method

5 Application to linear problem

6 Wrap-up

What you have to know

- IPM are state of the art algorithms for LP and more generally conic optimization problem
- That logarithmic barrier are a useful inner penalization method

What you really should know

- That Newton's algorithm can be applied with equality constraints
- What is the central path
- That IPM work with inner and outer optimization loop

Stochastic Gradient Method

V. Leclère (ENPC)

June 9th, 2023

Why should I bother to learn this stuff?

- Main algorithm principle for training machine learning model, and in particular deep neural network
- $\Longrightarrow$ useful for
  - understanding how the library train ML models
  - specialization in optimization
  - specialization in machine learning

The optimization problem ♡

We consider the following optimization problem

$$\underset{x\in\mathbb{R}^p}{\text{Min}} \qquad F(x) := \mathbb{E}\Big[f(x,\xi)\Big]$$

where ξ is a random variable.

Computing the gradient ♡

$$F(x) := \mathbb{E}\Big[f(x,\xi)\Big]$$

Under some regularity conditions (e.g. $f(\cdot,\xi)$ differentiable, $\frac{\partial f(x,\cdot)}{\partial x}$ Lipschitz, and ξ integrable) we have

$$\nabla F(x) = \mathbb{E}\Big[\frac{\partial f}{\partial x}(x,\xi)\Big]$$

This is obvious if ξ is finitely supported : $\mathrm{supp}(\xi) = \{\xi_i\}_{i\in[N]}$, and $p_i := \mathbb{P}(\xi = \xi_i)$,

$$\nabla F(x) = \frac{\partial}{\partial x}\Big(\sum_{i\in\ [N]} p_i f(x,\zeta)\Big) = \sum_{i\in\ [N]} p_i \frac{\partial}{\partial x} f(x,\zeta)$$

Standard continuous optimization method

Thus, we are looking at

$$\underset{x \in \mathbb{R}^p}{\text{Min}} \; F(x)$$

where F is a (strongly) convex differentiable function if $f(\cdot, \xi)$ is, and we know how to compute its gradient.

Thus, we should be able to solve our problem through the method presented in earlier courses:

- gradient algorithm
- conjugate gradient
- Newton / Quasi-Newton

Why bother with another (class of) algorithm ?

Computing the gradient is costly

For a given solution $x \in \mathbb{R}^p$ computing the gradient

$$\nabla F(x) = \mathbb{E}\Big[\frac{\partial f(x, \xi)}{\partial x}\Big]$$

is costly as it requires computing a multidimensional integral (if ξ admits a density), or a large sum.

Indeed, in most machine learning applications, we consider that ξ is uniformly distributed over the data (*empirical risk minimization*), thus computing the gradient requires a pass over every sample in the dataset.

Datasets of size $N > 10^6$ are common.

Estimating the gradient

Instead of using a true gradient

$$g^{(k)} = \nabla F(x^{(k)})$$

we can use a *statistical estimator* of the gradient

$$\hat{g}^{(k)} \rightsquigarrow g^{(k)} = \mathbb{E}\Big[\frac{\partial f(x^{(k)}, \xi)}{\partial x}\Big]$$

The most standard estimator being

$$\hat{g}^{(k)} = \frac{\partial f(x^{(k)}, \xi^{(k)})}{\partial x}$$

where $\xi^{(k)}$ is drawn randomly according to the law of ξ (i.e. it is a random datapoint).

Pros and Cons

Pros:

- computing $\hat{g}^{(k)} = \frac{\partial f(x^{(k)}, \xi^{(k)})}{\partial x}$ is really easy
- we do not need to spend lots of time early to get a precise gradient
- we can stop whenever we want (do not need a full pass on the data)

Cons:

- $\hat{g}^{(k)}$ is a noisy estimator of the gradient
- requires a new convergence theory
- $x^{(k+1)} := x^{(k+1)} - \alpha \hat{g}^{(k)}$ generally does not converge almost surely to the optimal solution as this makes a noisy trajectory

Noisy trajectory

- At optimality we should have $\nabla F(x^\sharp) = 0$
- It doesnot mean that $\frac{\partial f(x^\sharp, \xi^{(k)})}{\partial x}$ equals 0 !
- In particular there is no reason for $\hat{g}^{(k)}$ to be eventually small, only its expectation should be small!

- $\rightsquigarrow$ we generally use either:
  - decreasing step e.g. $\alpha^{(k)} = \dfrac{\alpha^{(0)}}{k}$
  - average points $\bar{x}^{(k)} = \dfrac{1}{k} \sum_{\kappa \le k} x^{(\kappa)}$

Mini-batch version

- $\hat{g}^{(k)} = \frac{\partial f(x^{(k)}, \xi^{(k)})}{\partial x}$ is an easy-to-compute but noisy estimator of the gradient
- $\hat{g}^{(k)} = \frac{1}{N} \sum_{i \in [N]} \frac{\partial f(x^{(k)}, \xi_i)}{\partial x}$ is a long (full batch) to compute but perfect estimator
- minibatch aims at a middle ground: randomly draw a sample S of realizations of $\boldsymbol{\xi}$, and use $\hat{g}^{(k)} = \frac{1}{|S|} \sum_{\xi \in S} \frac{\partial f(x^{(k)}, \xi)}{\partial x}$

Video explanation

Videos by Andrew Ng (Former Standford professor)

- https://www.youtube.com/watch?v=W9iWNJNFzQI&list=PLWbSa0uhIdsa6wpq9s_cKOP-PjeU0aIIu&index=24 (13')
- https://www.youtube.com/watch?v=l4lSUAcvHFs&list=PLWbSa0uhIdsa6wpq9s_cKOP-PjeU0aIIu&index=25(6')

Another video with numerical tricks to improve the convergence

- https://www.youtube.com/watch?v=kK8-jCCR4is&list=PLWbSa0uhIdsa6wpq9s_cKOP-PjeU0aIIu&index=23(10')

What you have to know

- That for a stochastic problem gradient step requires to compute an expectation
- That stochastic gradient do not compute the true gradient, but only an estimator of the gradient

What you really should know

- gradient algorithm (or more advanced version) is faster in term of number of iterations
- stochastic gradient needs more iteration, but each is faster

What you have to be able to do

- dive in the scientific litterature on the subject if you need to implement this type of algorithm