

Linear Programming by First-Order Methods

Stefano Gualandi

Università di Pavia, Dipartimento di Matematica

email: stefano.gualandi@unipv.it

twitter: [@famo2spaghi](https://twitter.com/famo2spaghi)

blog: <http://stegua.github.com>

Notation

We focus on Linear Programming (LP) problems written in standard form:

$$\text{(Primal)} \quad \min \quad \mathbf{c}^T \mathbf{x} \quad (1)$$

$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b} \quad (2)$$

$$\mathbf{x} \geq 0. \quad (3)$$

$$\text{(Dual)} \quad \max \quad \mathbf{b}^T \mathbf{y} \quad (4)$$

$$\text{s.t.} \quad \mathbf{A}^T \mathbf{y} \leq \mathbf{c} \quad (5)$$

$$\mathbf{y} \text{ free.} \quad (6)$$

where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$.

Notation

We focus on Linear Programming (LP) problems written in standard form:

$$\begin{array}{llll}
 \text{(Primal)} & \min & \mathbf{c}^T \mathbf{x} & (1) \\
 & \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} & (2) \\
 & & \mathbf{x} \geq 0. & (3) \\
 \text{(Dual)} & \max & \mathbf{b}^T \mathbf{y} & (4) \\
 & \text{s.t.} & \mathbf{A}^T \mathbf{y} \leq \mathbf{c} & (5) \\
 & & \mathbf{y} \text{ free.} & (6)
 \end{array}$$

where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$.

Convex function over a convex set

In LP, **local optimality** implies **global optimality**, since we are minimizing a **convex function** over a **convex set**.

Size classification of optimization methods

Nesterov categorizes optimization methods (AHLL21):

- **Factorization methods:** They use factorization to solve exactly linear system subproblems (the **enemy** is **matrix inversion**). They can handle to **medium size** problems.

Size classification of optimization methods

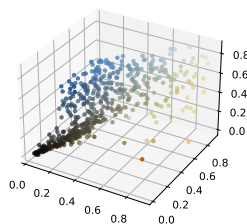
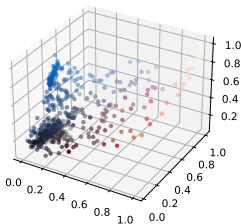
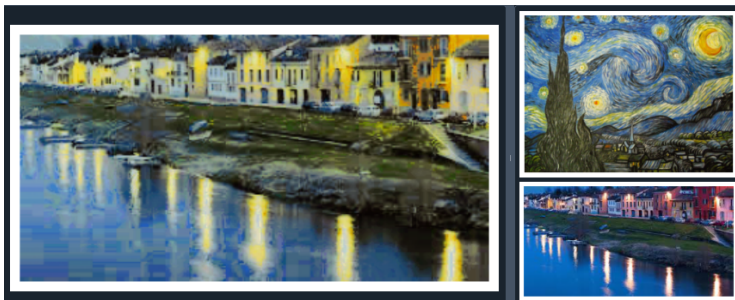
Nesterov categorizes optimization methods (AHLL21):

- **Factorization methods:** They use factorization to solve exactly linear system subproblems (the **enemy** is **matrix inversion**). They can handle to **medium size** problems.
- **Matrix-vector methods:** The most expensive operation is a matrix-vector multiplication. The memory footprint is the same of the input data. They can handle **very large size** problems.

Color Transfer by Optimal Transport: Very large LP

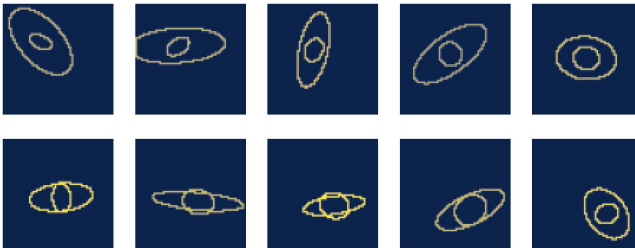


Color Transfer by Optimal Transport: Very large LP



Motivations: Solution of very large-scale LPs

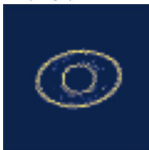
While for **medium size** instances commercial solvers are the best choice in terms of trade-off between runtime and solution accuracy. However, for modern **very-large size** instances from Machine Learning (but not only!!!), we have to trade off among runtime, accuracy, and memory.



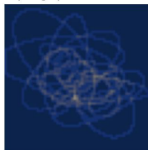
Motivations: Solution of very large-scale LPs

While for **medium size** instances commercial solvers are the best choice in terms of trade-off between runtime and solution accuracy. However, for modern **very-large size** instances from Machine Learning (but not only!!!), we have to trade off among runtime, accuracy, and memory.

Wasserstein
Ran in 153.95 s
Opt. gap = 0.0 %



Euclidean
Ran in 0.04 s
Opt. gap = 32.035 %



Entropic $\epsilon = 3.5$
Ran in 40.65 s
Opt. gap = 0.893 %



Entropic $\epsilon = 10$
Ran in 8.4 s
Opt. gap = 2.549 %



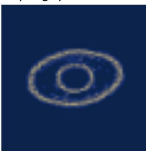
MAAIPM
Ran in 137.87 s
Opt. gap = 0.058 %



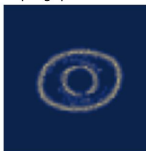
Farthest H
Ran in 4.23 s
Opt. gap = 4.077 %



PairRand
Ran in 5.47 s
Opt. gap = 2.082 %



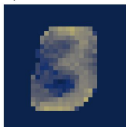
PairFarthest
Ran in 4.33 s
Opt. gap = 0.459 %



Motivations: Solution of very large-scale LPs

While for **medium size** instances commercial solvers are the best choice in terms of trade-off between runtime and solution accuracy. However, for modern **very-large size** instances from Machine Learning (but not only!!!), we have to trade off among runtime, accuracy, and memory.

MAAIPM
Ran in 535.33 s
Opt. value = 840.0



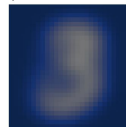
Euclidean
Ran in 0.28 s
Opt. value = 1011.5



Entropic $\varepsilon = 1$
Ran in 23.66 s
Opt. value = 921.0



Entropic $\varepsilon = 10$
Ran in 1.43 s
Opt. value = 1206.1



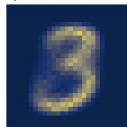
Iterative H
Ran in 11.48 s
Opt. value = 929.6



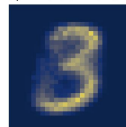
Farthest H
Ran in 101.22 s
Opt. value = 939.0



PairRand
Ran in 7.94 s
Opt. value = 924.5



PairFarthest
Ran in 8.3 s
Opt. value = 929.9



Theorem 1 (Weak Duality)

If x is a feasible solution to the primal problem and y is a feasible solution to the dual problem, then

$$y^T b \leq c^T x. \quad (7)$$

Theorem 1 (Weak Duality)

If $\mathbf{x}$ is a feasible solution to the primal problem and $\mathbf{y}$ is a feasible solution to the dual problem, then

$$\mathbf{y}^T \mathbf{b} \leq \mathbf{c}^T \mathbf{x}. \quad (7)$$

Corollary 2

- (a) *If the optimal solution in the primal is $-\infty$, then the dual problem must be infeasible.*
- (b) *If the optimal solution in the dual is $+\infty$, then the primal problem must be infeasible.*

Theorem 1 (Weak Duality)

If $\mathbf{x}$ is a feasible solution to the primal problem and $\mathbf{y}$ is a feasible solution to the dual problem, then

$$\mathbf{y}^T \mathbf{b} \leq \mathbf{c}^T \mathbf{x}. \quad (7)$$

Corollary 2

- (a) *If the optimal solution in the primal is $-\infty$, then the dual problem must be infeasible.*
- (b) *If the optimal solution in the dual is $+\infty$, then the primal problem must be infeasible.*

Corollary 3

Let $\mathbf{x}$ and $\mathbf{y}$ be feasible solutions to the primal and the dual, respectively, and suppose that $\mathbf{y}^T \mathbf{b} = \mathbf{c}^T \mathbf{x}$. Then, $\mathbf{x}$ and $\mathbf{y}$ are optimal solutions to the primal and the dual, respectively.

Theorem 4 (Strong Duality)

If a linear programming problem has an optimal solution, so does its dual, and the respective optimal costs are equal.

Theorem 4 (Strong Duality)

If a linear programming problem has an optimal solution, so does its dual, and the respective optimal costs are equal.

Theorem 5 (Complementary Slackness)

Let $\mathbf{x}$ and $\mathbf{y}$ be feasible solutions to the primal and dual problem, respectively. The vectors $\mathbf{x}$ and $\mathbf{y}$ are optimal solutions for the two respective problems if and only if:

$$y_i(\mathbf{a}^T \mathbf{x} - b_i) = 0 \quad \forall i = 1, \dots, m \quad (8)$$

$$(c_j - \mathbf{y}^T \mathbf{A}_j)x_j = 0 \quad \forall j = 1, \dots, n. \quad (9)$$

Solution Algorithms for LPs

- Simplex algorithms (primal, dual, primal-dual, [network-simplex](#))

Solution Algorithms for LPs

- Simplex algorithms (primal, dual, primal-dual, [network-simplex](#))
- Interior point algorithms (aka barrier algorithms)

Solution Algorithms for LPs

- Simplex algorithms (primal, dual, primal-dual, [network-simplex](#))
- Interior point algorithms (aka barrier algorithms)
- **First-order methods**

Simplex algorithm

TODO: sketch with iPad: pivoting in and out of basis

Interior Point algorithm

BASIC IDEA: To solve a *Constrained Linear Problem*, we solve a *Nonlinear Problem*, called the **Barrier Problem**.

Interior Point algorithm

BASIC IDEA: To solve a *Constrained Linear Problem*, we solve a *Nonlinear Problem*, called the **Barrier Problem**.

We are interested in solve the pair of Primal and Dual problems:

$$\max \quad c^T x$$

$$\text{s.t.} \quad Ax + w = b,$$

$$x, w \geq 0.$$

$$\min \quad b^T y$$

$$\text{s.t.} \quad A^T y - z = c,$$

$$y, z \geq 0.$$

QUESTION: Could we remove the nonnegative constraints and add a suitable penalizing term to the objective function?

Interior Point algorithm

BASIC IDEA: To solve a *Constrained Linear Problem*, we solve a *Nonlinear Problem*, called the **Barrier Problem**.

We are interested in solve the pair of Primal and Dual problems:

$$\max \quad c^T x$$

$$\text{s.t.} \quad Ax + w = b,$$

$$x, w \geq 0.$$

$$\min \quad b^T y$$

$$\text{s.t.} \quad A^T y - z = c,$$

$$y, z \geq 0.$$

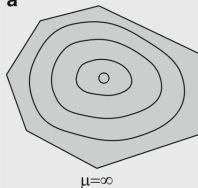
QUESTION: Could we remove the nonnegative constraints and add a suitable penalizing term to the objective function?

Using the parameter $\mu > 0$ and the $\log()$ function, we define the **Barrier Problem**:

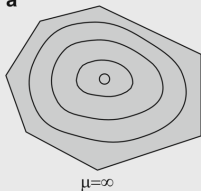
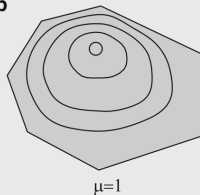
$$\max \quad c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i)$$

$$\text{s.t.} \quad Ax + w = b.$$

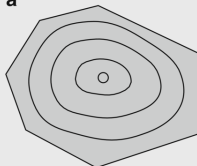
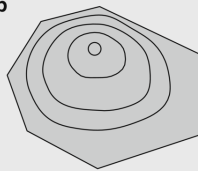
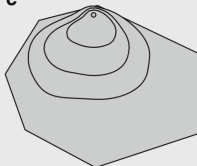
Intuition in $\mathbb{R}^2$: The Central Path

a

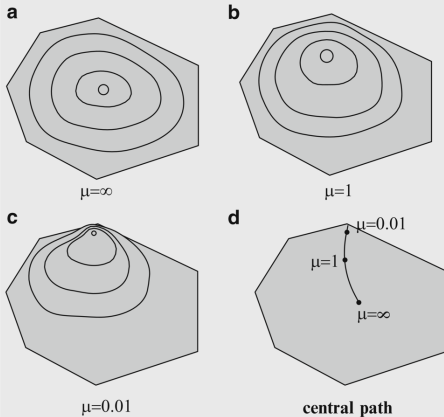
Intuition in $\mathbb{R}^2$: **The Central Path**

a**b**

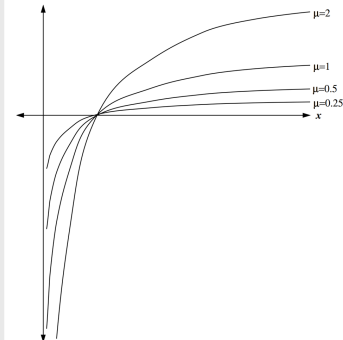
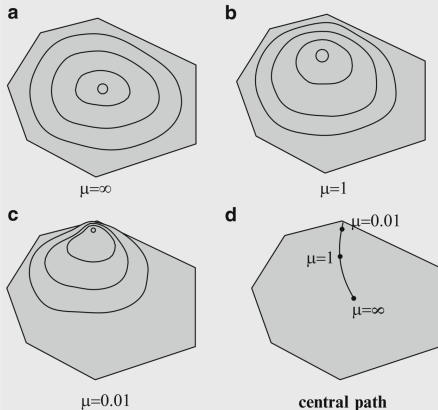
Intuition in $\mathbb{R}^2$: The Central Path

a $\mu=\infty$ **b** $\mu=1$ **c** $\mu=0.01$

Intuition in $\mathbb{R}^2$: The Central Path



Intuition in $\mathbb{R}^2$: The Central Path



Barrier Problem plus Lagrangian

Given the **Barrier Problem**:

$$\begin{aligned} \max \quad & c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i) \\ \text{s.t.} \quad & Ax + w = b. \end{aligned}$$

Barrier Problem plus Lagrangian

Given the **Barrier Problem**:

$$\begin{aligned} \max \quad & c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i) \\ \text{s.t.} \quad & Ax + w = b. \end{aligned}$$

We introduce the following Lagrange function:

$$L(x, w, y) = c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i) + y^T(b - Ax - w)$$

Barrier Problem plus Lagrangian

Given the **Barrier Problem**:

$$\begin{aligned} \max \quad & c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i) \\ \text{s.t.} \quad & Ax + w = b. \end{aligned}$$

We introduce the following Lagrange function:

$$L(x, w, y) = c^T x + \mu \sum_{j=1}^n \log(x_j) + \mu \sum_{i=1}^m \log(w_i) + y^T (b - Ax - w)$$

Forcing the first-order optimality conditions:

$$\begin{aligned} \frac{\partial L}{\partial x_j} &= c_j + \mu \frac{1}{x_j} - \sum_{i=1}^m a_{ij} y_i = 0, & j = 1, \dots, n \\ \frac{\partial L}{\partial w_i} &= \mu \frac{1}{w_i} - y_i = 0, & i = 1, \dots, m \\ \frac{\partial L}{\partial y_i} &= b_i - \sum_{j=1}^n a_{ij} x_j - w_i = 0, & i = 1, \dots, m. \end{aligned} \tag{10}$$

Barrier Problem plus Lagrangian

Forcing the first-order optimality conditions:

$$\begin{aligned}
 \frac{\partial L}{\partial x_j} &= c_j + \mu \frac{1}{x_j} - \sum_{i=1}^m a_{ij} y_i = 0, & j = 1, \dots, n \\
 \frac{\partial L}{\partial w_i} &= \mu \frac{1}{w_i} - y_i = 0, & i = 1, \dots, m \\
 \frac{\partial L}{\partial y_i} &= b_i - \sum_{j=1}^n a_{ij} x_j - w_i = 0, & i = 1, \dots, m.
 \end{aligned} \tag{11}$$

Barrier Problem plus Lagrangian

Forcing the first-order optimality conditions:

$$\begin{aligned}
 \frac{\partial L}{\partial x_j} &= c_j + \mu \frac{1}{x_j} - \sum_{i=1}^m a_{ij} y_i = 0, & j = 1, \dots, n \\
 \frac{\partial L}{\partial w_i} &= \mu \frac{1}{w_i} - y_i = 0, & i = 1, \dots, m \\
 \frac{\partial L}{\partial y_i} &= b_i - \sum_{j=1}^n a_{ij} x_j - w_i = 0, & i = 1, \dots, m.
 \end{aligned} \tag{11}$$

Using the notation $X = \text{diag}(x_1, \dots, x_n)$, $W = \text{diag}(w_1, \dots, w_m)$, and e is a vector of all 1 of appropriate size,

Barrier Problem plus Lagrangian

Forcing the first-order optimality conditions:

$$\begin{aligned}
 \frac{\partial L}{\partial x_j} &= c_j + \mu \frac{1}{x_j} - \sum_{i=1}^m a_{ij} y_i = 0, & j = 1, \dots, n \\
 \frac{\partial L}{\partial w_i} &= \mu \frac{1}{w_i} - y_i = 0, & i = 1, \dots, m \\
 \frac{\partial L}{\partial y_i} &= b_i - \sum_{j=1}^n a_{ij} x_j - w_i = 0, & i = 1, \dots, m.
 \end{aligned} \tag{11}$$

Using the notation $X = \text{diag}(x_1, \dots, x_n)$, $W = \text{diag}(w_1, \dots, w_m)$, and e is a vector of all 1 of appropriate size, we can rewrite the conditions in matrix notation

$$\begin{aligned}
 A^T y - \mu X^{-1} e &= c \\
 y &= \mu W^{-1} e \\
 Ax + w &= b.
 \end{aligned}$$

Barrier Problem plus Lagrangian

Forcing the first-order optimality conditions:

$$\begin{aligned}
 \frac{\partial L}{\partial x_j} &= c_j + \mu \frac{1}{x_j} - \sum_{i=1}^m a_{ij} y_i = 0, & j = 1, \dots, n \\
 \frac{\partial L}{\partial w_i} &= \mu \frac{1}{w_i} - y_i = 0, & i = 1, \dots, m \\
 \frac{\partial L}{\partial y_i} &= b_i - \sum_{j=1}^n a_{ij} x_j - w_i = 0, & i = 1, \dots, m.
 \end{aligned} \tag{11}$$

Using the notation $X = \text{diag}(x_1, \dots, x_n)$, $W = \text{diag}(w_1, \dots, w_m)$, and e is a vector of all 1 of appropriate size, we can rewrite the conditions in matrix notation

$$\begin{aligned}
 A^T y - \mu X^{-1} e &= c \\
 y &= \mu W^{-1} e \\
 Ax + w &= b.
 \end{aligned}$$

Adding the extra vector $z = \mu X^{-1} e$, we can rewrite as

$$Ax + w = b \tag{12}$$

$$A^T y - z = c \tag{13}$$

$$z = \mu X^{-1} e \tag{14}$$

$$y = \mu W^{-1} e \tag{15}$$

$$XZ = \mu e$$

$$WY = \mu e$$

Writing the last two equations component-wise, we get the μ -complementary conditions:

$$\begin{aligned} x_j z_j &= \mu, & j = 1, \dots, n \\ y_i w_i &= \mu, & i = 1, \dots, m. \end{aligned}$$

Note that for $\mu = 0$ these are exactly the **Slackness Complementary** conditions.

Writing the last two equations component-wise, we get the μ -complementary conditions:

$$\begin{aligned}x_j z_j &= \mu, & j = 1, \dots, n \\ y_i w_i &= \mu, & i = 1, \dots, m.\end{aligned}$$

Note that for $\mu = 0$ these are exactly the **Slackness Complementary** conditions.

We have $2(n + m)$ equations with $2(n + m)$ unknowns. If the equations were linear, “*mister Gauss would do the job*”.

Writing the last two equations component-wise, we get the μ -complementary conditions:

$$\begin{aligned} x_j z_j &= \mu, & j &= 1, \dots, n \\ y_i w_i &= \mu, & i &= 1, \dots, m. \end{aligned}$$

Note that for $\mu = 0$ these are exactly the **Slackness Complementary** conditions.

We have $2(n + m)$ equations with $2(n + m)$ unknowns. If the equations were linear, “*mister Gauss would do the job*”.

This little nonlinearity makes Linear Programming nontrivial.

Basic definitions (Roc76)

Definition 6 (Convex function)

An extended real valued function $f : X \rightarrow [-\infty, +\infty]$ is said to be **convex** if and only if its **epigraph**

$$\text{epif} := \{(x, \lambda) \in X \times \mathbb{R} \mid \lambda \geq f(x)\}$$

is a convex set, that is, if when $\lambda \geq f(x)$, $\mu \geq f(y)$, and $t \in [0, 1]$, we have:

$$f(tx + (1 - t)y) \leq t\lambda + (1 - t)\mu.$$

Definition 7 (Proper convex function)

A convex function f is **proper** if it is not identically $+\infty$ and nowhere $-\infty$: in this case, it is convex if and only if, for all $x, y \in X$ and $t \in [0, 1]$

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$$

Definition 8 (Strictly convex function)

A convex function f is **strictly convex** if whenever $x \neq y$ and $0 < t < 1$, we have

$$f(tx + (1 - t)y) < tf(x) + (1 - t)f(y).$$

Basic definitions

Definition 9 (Lower semi-continuous function (l.s.c))

An extended real valued function $f : X \rightarrow [-\infty, +\infty]$ is said to be **lower semi-continuous (l.s.c.)** if, for all $x \in X$, if $x_n \rightarrow x$, then

$$f(x) \leq \liminf_{n \rightarrow \infty} f(x_n).$$

Basic definitions

Definition 9 (Lower semi-continuous function (l.s.c))

An extended real valued function $f : X \rightarrow [-\infty, +\infty]$ is said to be **lower semi-continuous (l.s.c.)** if, for all $x \in X$, if $x_n \rightarrow x$, then

$$f(x) \leq \liminf_{n \rightarrow \infty} f(x_n).$$

Example 10 (Indicator function)

The *characteristic function* or *indicator function* of a set C defined as

$$\delta_C(x) = \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{otherwise,} \end{cases}$$

is *convex*, l.s.c., and *proper* when C is convex, closed, and non-empty.

Basic definitions

Definition 11 (Subgradient)

Given a convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$, its **subgradient** is defined as

$$\partial f(x) := \{p \in X \mid f(y) \geq f(x) + \langle p, y - x \rangle, \forall y \in X\}.$$

Basic definitions

Definition 11 (Subgradient)

Given a convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$, its **subgradient** is defined as

$$\partial f(x) := \{p \in X \mid f(y) \geq f(x) + \langle p, y - x \rangle, \forall y \in X\}.$$

Proposition 3.1 (Minimizer of non-smooth convex function)

$x \in X$ is a **global minimizer** of f if and only if $0 \in \partial f(x)$.

Basic definitions

Definition 11 (Subgradient)

Given a convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$, its **subgradient** is defined as

$$\partial f(x) := \{p \in X \mid f(y) \geq f(x) + \langle p, y - x \rangle, \forall y \in X\}.$$

Proposition 3.1 (Minimizer of non-smooth convex function)

$x \in X$ is a **global minimizer** of f if and only if $0 \in \partial f(x)$.

Definition 12 (Strongly convex or μ -convex)

A convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$ is **strongly convex** if

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

Basic definitions

Definition 11 (Subgradient)

Given a convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$, its **subgradient** is defined as

$$\partial f(x) := \{p \in X \mid f(y) \geq f(x) + \langle p, y - x \rangle, \forall y \in X\}.$$

Proposition 3.1 (Minimizer of non-smooth convex function)

$x \in X$ is a **global minimizer** of f if and only if $0 \in \partial f(x)$.

Definition 12 (Strongly convex or μ -convex)

A convex, extended real valued, l.s.c. function $f : X \rightarrow [-\infty, +\infty]$ is **strongly convex** if

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

REMARK: If f is *strongly convex* and x is a minimizer, then we have

$$f(y) \geq f(x) + \frac{\mu}{2} \|y - x\|^2, \quad \forall y \in X.$$

Basic definitions

Definition 13 (Convex conjugate)

To any function $f : X \rightarrow [-\infty, +\infty]$ we can associate the [Legendre-Fenchel conjugate](#) (or [convex conjugate](#))

$$f^*(y) = \sup_{x \in X} \langle y, x \rangle - f(x)$$

which is convex and semi-continuous. If f is convex and l.s.c., we have $f^{**} = f$.

Basic definitions

Definition 13 (Convex conjugate)

To any function $f : X \rightarrow [-\infty, +\infty]$ we can associate the [Legendre-Fenchel conjugate](#) (or [convex conjugate](#))

$$f^*(y) = \sup_{x \in X} \langle y, x \rangle - f(x)$$

which is convex and semi-continuous. If f is convex and l.s.c., we have $f^{**} = f$.

Example 14

The convex conjugate of the affine function $f(x) = \langle a, x \rangle - b$ is

$$f^*(y) = \begin{cases} b, & \text{if } y = a, \\ +\infty & \text{otherwise.} \end{cases}$$

Basic definitions

Definition 13 (Convex conjugate)

To any function $f : X \rightarrow [-\infty, +\infty]$ we can associate the **Legendre-Fenchel conjugate** (or **convex conjugate**)

$$f^*(y) = \sup_{x \in X} \langle y, x \rangle - f(x)$$

which is convex and semi-continuous. If f is convex and l.s.c., we have $f^{**} = f$.

Example 14

The convex conjugate of the affine function $f(x) = \langle a, x \rangle - b$ is

$$f^*(y) = \begin{cases} b, & \text{if } y = a, \\ +\infty & \text{otherwise.} \end{cases}$$

Example 15

The convex conjugate of the function $f(x) = e^x$ is

$$f^*(y) = \begin{cases} y \log(y) - y, & \text{if } y > 0, \\ 0 & y = 0. \end{cases}$$

Definition 16 (Proximal map or proximal operator)

If f is convex, proper, and l.s.c. then for any x there is a unique minimizer $\bar{y}$ to the strongly convex problem

$$\bar{y} := \text{prox}_{\tau f}(x) = \arg \min_{y \in X} f(y) + \frac{1}{2\tau} \|y - x\|^2$$

Consider the minimization problem

$$\min_{x \in X} f(Kx) + g(x)$$

where

$$f : Y \rightarrow [-\infty, +\infty] \text{ and } g : X \rightarrow [-\infty, +\infty]$$

are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Consider the minimization problem

$$\min_{x \in X} f(Kx) + g(x)$$

where

$$f : Y \rightarrow [-\infty, +\infty] \text{ and } g : X \rightarrow [-\infty, +\infty]$$

are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Then, we can write

$$\min_{x \in X} f(Kx) + g(x) = \min_{x \in X} \sup_{y \in Y} \langle y, Kx \rangle - f^*(y) + g(x).$$

Consider the minimization problem

$$\min_{x \in X} f(Kx) + g(x)$$

where

$$f : Y \rightarrow [-\infty, +\infty] \text{ and } g : X \rightarrow [-\infty, +\infty]$$

are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Then, we can write

$$\min_{x \in X} f(Kx) + g(x) = \min_{x \in X} \sup_{y \in Y} \langle y, Kx \rangle - f^*(y) + g(x).$$

Under mild conditions on f and g , we can write the [Fenchel-Rockafellar dual problem](#):

$$\min_{x \in X} f(Kx) + g(x) = \min_{x \in X} \sup_{y \in Y} \langle y, Kx \rangle - f^*(y) + g(x) \quad (16)$$

$$= \max_{y \in Y} \inf_{x \in X} \langle y, Kx \rangle - f^*(y) + g(x) \quad (17)$$

$$= \max_{y \in Y} -f^*(y) - g^*(-K^*y). \quad (18)$$

Arrow, Hurwicz, and Uzawa 1958

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Arrow, Hurwicz, and Uzawa 1958

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator. Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

Arrow, Hurwicz, and Uzawa 1958

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator. Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

which can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using the iterates (e.g., see (CP16)):

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k) = \arg \min_{x \in X} g(x) + \frac{1}{2\tau} \left\| (x^k - \tau K^* y^k) - x \right\|^2, \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K x^{k+1}) = \arg \min_{y \in Y} f^*(y) + \frac{1}{2\sigma} \left\| y - (y^k + \sigma K x^{k+1}) \right\|^2. \end{aligned}$$

Arrow, Hurwicz, and Uzawa 1958

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator. Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

which can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using the iterates (e.g., see (CP16)):

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k) = \arg \min_{x \in X} g(x) + \frac{1}{2\tau} \left\| (x^k - \tau K^* y^k) - x \right\|^2, \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K x^{k+1}) = \arg \min_{y \in Y} f^*(y) + \frac{1}{2\sigma} \left\| y - (y^k + \sigma K x^{k+1}) \right\|^2. \end{aligned}$$

QUESTION 1: Does these iterates converge to a saddle-point?

Arrow, Hurwicz, and Uzawa 1958

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator. Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

which can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using the iterates (e.g., see (CP16)):

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k) = \arg \min_{x \in X} g(x) + \frac{1}{2\tau} \left\| (x^k - \tau K^* y^k) - x \right\|^2, \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K x^{k+1}) = \arg \min_{y \in Y} f^*(y) + \frac{1}{2\sigma} \left\| y - (y^k + \sigma K x^{k+1}) \right\|^2. \end{aligned}$$

QUESTION 1: Does these iterates converge to a saddle-point?

QUESTION 2: If yes, which is the convergence rate?

Chambolle and Pock, 2011 (with applications to imaging)

The *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k), \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K(x^{k+1} + \theta(x^{k+1} - x^k))). \end{aligned}$$

Note that for $\theta = 0$ we get the previous slide scheme (special case of the proximal algorithm (Roc76)).

Chambolle and Pock, 2011 (with applications to imaging)

The *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k), \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K(x^{k+1} + \theta(x^{k+1} - x^k))). \end{aligned}$$

Note that for $\theta = 0$ we get the previous slide scheme (special case of the proximal algorithm (Roc76)).

If $\theta = 1$ and $\tau\sigma \|K\|^2 < 1$, the given iterates converge to a saddle-point (x^*, y^*) .

Chambolle and Pock, 2011 (with applications to imaging)

The *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k), \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K(x^{k+1} + \theta(x^{k+1} - x^k))). \end{aligned}$$

Note that for $\theta = 0$ we get the previous slide scheme (special case of the proximal algorithm (Roc76)).

If $\theta = 1$ and $\tau\sigma \|K\|^2 < 1$, the given iterates converge to a saddle-point (x^*, y^*) .

We refer to this algorithm as the **Primal-Dual Hybrid Gradient (PDHG) method** (CP11; PC11).

Chambolle and Pock, 2011 (with applications to imaging)

The *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k), \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K(x^{k+1} + \theta(x^{k+1} - x^k))). \end{aligned}$$

Note that for $\theta = 0$ we get the previous slide scheme (special case of the proximal algorithm (Roc76)).

If $\theta = 1$ and $\tau\sigma \|K\|^2 < 1$, the given iterates converge to a saddle-point (x^*, y^*) .

We refer to this algorithm as the **Primal-Dual Hybrid Gradient (PDHG) method** (CP11; PC11).

QUESTION 3: Can we customize this algorithm to Linear Programming?

Chambolle and Pock, 2011 (with applications to imaging)

The *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + g(x) - f^*(y)$$

can be solved by a descent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\begin{aligned} x^{k+1} &= \text{prox}_{\tau g}(x^k - \tau K^* y^k), \\ y^{k+1} &= \text{prox}_{\sigma f^*}(y^k + \sigma K(x^{k+1} + \theta(x^{k+1} - x^k))). \end{aligned}$$

Note that for $\theta = 0$ we get the previous slide scheme (special case of the proximal algorithm (Roc76)).

If $\theta = 1$ and $\tau\sigma \|K\|^2 < 1$, the given iterates converge to a saddle-point (x^*, y^*) .

We refer to this algorithm as the **Primal-Dual Hybrid Gradient (PDHG) method** (CP11; PC11).

QUESTION 3: Can we customize this algorithm to Linear Programming?

QUESTION 4: Can we customize this algorithm to Optimal Transport LPs?

Primal-Dual algorithm for solving LP

TODO: sketch with iPad

Primal-Dual algorithm for solving LP

Given an LP in standard form:

$$\text{(Primal)} \quad \min \quad \mathbf{c}^\top \mathbf{x} \quad (19)$$

$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b} \quad (20)$$

$$\mathbf{x} \geq 0. \quad (21)$$

We reformulate the LP as the minimax saddle point problem:

$$\min_{\mathbf{x} \in X} \max_{\mathbf{y} \in Y} \langle \mathbf{y}, \mathbf{Ax} \rangle + \langle \mathbf{c}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{b} \rangle$$

Primal-Dual algorithm for solving LP

Given an LP in standard form:

$$\text{(Primal)} \quad \min \quad c^T x \quad (19)$$

$$\text{s.t.} \quad Ax = b \quad (20)$$

$$x \geq 0. \quad (21)$$

We reformulate the LP as the minimax saddle point problem:

$$\min_{x \in X} \max_{y \in Y} \langle y, Ax \rangle + \langle c, x \rangle - \langle y, b \rangle$$

which can be solved by a descendent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$x^{k+1} = \text{prox}_{\tau g}(x^k - \tau A^T y^k) = \text{proj}_{x \geq 0}(x^k - \tau(c - A^T y^k))$$

Primal-Dual algorithm for solving LP

Given an LP in standard form:

$$\text{(Primal)} \quad \min \quad \mathbf{c}^\top \mathbf{x} \quad (19)$$

$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b} \quad (20)$$

$$\mathbf{x} \geq 0. \quad (21)$$

We reformulate the LP as the minimax saddle point problem:

$$\min_{\mathbf{x} \in X} \max_{\mathbf{y} \in Y} \langle \mathbf{y}, \mathbf{Ax} \rangle + \langle \mathbf{c}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{b} \rangle$$

which can be solved by a descendent in the variable $\mathbf{x}$ with a step size $\tau > 0$ and an ascent in the variable $\mathbf{y}$ with step size $\sigma > 0$, using an **over relaxation** step as follows

$$\mathbf{x}^{k+1} = \text{prox}_{\tau \mathbf{g}}(\mathbf{x}^k - \tau \mathbf{A}^\top \mathbf{y}^k) = \text{proj}_{\mathbf{x} \geq 0}(\mathbf{x}^k - \tau(\mathbf{c} - \mathbf{A}^\top \mathbf{y}^k))$$

$$\mathbf{y}^{k+1} = \text{prox}_{\sigma f^*}(\mathbf{y}^k + \sigma \mathbf{A}(\mathbf{x}^{k+1} + \theta(\mathbf{x}^{k+1} - \mathbf{x}^k))) = \mathbf{y}^k + \sigma(\mathbf{b} - \mathbf{A}(\mathbf{x}^{k+1} + \theta(\mathbf{x}^{k+1} - \mathbf{x}^k)))$$

where

$$\tau = \frac{\nu}{\omega}, \quad \sigma = \omega \nu, \quad \text{with } \nu, \omega \in (0, \infty)$$

Primal-Dual algorithm for solving LP

Given an LP in standard form:

$$\text{(Primal)} \quad \min \quad c^T x \quad (19)$$

$$\text{s.t.} \quad Ax = b \quad (20)$$

$$x \geq 0. \quad (21)$$

We reformulate the LP as the minimax saddle point problem:

$$\min_{x \in X} \max_{y \in Y} \langle y, Ax \rangle + \langle c, x \rangle - \langle y, b \rangle$$

which can be solved by a descendent in the variable x with a step size $\tau > 0$ and an ascent in the variable y with step size $\sigma > 0$, using an **over relaxation** step as follows

$$x^{k+1} = \text{prox}_{\tau g}(x^k - \tau A^T y^k) = \text{proj}_{x \geq 0}(x^k - \tau(c - A^T y^k))$$

$$y^{k+1} = \text{prox}_{\sigma f^*}(y^k + \sigma A(x^{k+1} + \theta(x^{k+1} - x^k))) = y^k + \sigma(b - A(x^{k+1} + \theta(x^{k+1} - x^k)))$$

where

$$\tau = \frac{\nu}{\omega}, \quad \sigma = \omega\nu, \quad \text{with } \nu, \omega \in (0, \infty)$$

REMARK: The algorithm converges whenever $\nu < \frac{1}{\|A\|_2}$.

Basic implementation in Julia (with $\theta = 1$)

```

 8  function PDHG(c, A, b, iter,  $\sigma_0$  )
 9       $x_0$  = [0.0 for i  $\in$  c]
10       $y_0$  = [0.0 for j  $\in$  b]
11       $\sigma$ ,  $\tau$  =  $\sigma_0$ ,  $\sigma_0$ 
12       $x_1$ ,  $y_1$  =  $x_0$ ,  $y_0$ 
13      for k  $\in$  1:iter
14           $x_1$  =  $x_0$  -  $\tau$ *(c - transpose(A)* $y_0$ )
15          for i  $\in$  1:n
16               $x_1[i]$  = max(0.0,  $x_1[i]$ )
17          end
18           $y_1$  =  $y_0$  +  $\sigma$ *(b - A*(2* $x_1$  -  $x_0$ ))
19           $x_0$ ,  $y_0$  =  $x_1$ ,  $y_1$ 
20          # Missing: better stopping criteria
21      end
22      return  $x_1$ 
23  end

```

Live Demo in Julia + Pluto

Primal-Dual LP by Google Research Team

Applegate et al. *Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient*. Arxiv, 2021.

The authors propose an enhancement of PDHG for LP problems, called **PDLP**, which has the following features:

- A “smart” **restarting** strategy, to avoid *tailing-off* effects.

Primal-Dual LP by Google Research Team

Applegate et al. *Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient*. Arxiv, 2021.

The authors propose an enhancement of PDHG for LP problems, called **PDLP**, which has the following features:

- A “smart” **restarting** strategy, to avoid *tailing-off* effects.
- A **scaling** technique based on a *diagonal preconditioner*, which has no effect on OT problems. This has effect on $\|A\|_2$.

Primal-Dual LP by Google Research Team

Applegate et al. *Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient*. Arxiv, 2021.

The authors propose an enhancement of PDHG for LP problems, called **PDLP**, which has the following features:

- A “smart” **restarting** strategy, to avoid *tailing-off* effects.
- A **scaling** technique based on a *diagonal preconditioner*, which has no effect on OT problems. This has effect on $\|A\|_2$.
- An adaptive rule for **changing the primal weight** ω

Primal-Dual LP by Google Research Team

Applegate et al. *Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient*. Arxiv, 2021.

The authors propose an enhancement of PDHG for LP problems, called **PDLP**, which has the following features:

- A “smart” **restarting** strategy, to avoid *tailing-off* effects.
- A **scaling** technique based on a *diagonal preconditioner*, which has no effect on OT problems. This has effect on $\|A\|_2$.
- An adaptive rule for **changing the primal weight** ω
- An adaptive step-size choice which **dynamically changes the step size** η (using the update of ω)

Primal-Dual LP by Google Research Team

Applegate et al. *Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient*. Arxiv, 2021.

The authors propose an enhancement of PDHG for LP problems, called **PDLP**, which has the following features:

- A “smart” **restarting** strategy, to avoid *tailing-off* effects.
- A **scaling** technique based on a *diagonal preconditioner*, which has no effect on OT problems. This has effect on $\|A\|_2$.
- An adaptive rule for **changing the primal weight** ω
- An adaptive step-size choice which **dynamically changes the step size** η (using the update of ω)
- **Presolving** the input LP

Results from Applegate et al.

Table 13: Performance statistics: MIP Relaxations Improvements Tol 1E-04

Experiment	Solved count	KKT passes SGM10	Solve time secs SGM10
PDHG	113	38958.0	1088.3
+ restarts	140	29739.6	770.4
+ scaling	221	14801.5	313.6
+ primal weight	315	7228.1	110.8
+ step size	332	3615.3	67.6
+ presolve (= PDLP)	349	3058.3	42.1

Table 14: Performance statistics: MIP Relaxations Improvements Tol 1E-08

Experiment	Solved count	KKT passes SGM10	Solve time secs SGM10
PDHG	48	68588.8	2232.6
+ restarts	101	47301.1	1284.0
+ scaling	162	25985.7	595.7
+ primal weight	223	18273.4	331.3
+ step size	269	10091.0	181.8
+ presolve (= PDLP)	283	9773.3	131.5

Results from Applegate et al.

Table 1: Instances from MIP **Relaxations** (top) and LP **benchmark** (bottom) where PDLP is within a factor of 2 of the best of all Gurobi methods. Time to solve in seconds.

Instance	PDLP	Gurobi Barrier	Gurobi Primal Simp.	Gurobi Dual Simp.
ex9	1.6	102.6	181.3	47.6
genus-sym-g62-2	2.1	10.7	6.7	33.2
highschool1-aigio	72.6	243.8	>3600	>3600
neos-578379	1.4	0.7	1.7	1.8
rwth-timetable	1870.3	>3600	>3600	>3600
ex10	4.9	63.1	16.8	7.9
nug08-3rd	2.2	3.2	2219.2	24.1
savshed1	35.9	25.9	56.0	261.3

Research Question 1: Different Saddle-point formulations

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + \langle c, x \rangle - \langle y, b \rangle$$

If f is the indicator function, we get the Lagrangian relaxation and the previous saddle point formulation of LP.

Research Question 1: Different Saddle-point formulations

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator. Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + \langle c, x \rangle - \langle y, b \rangle$$

If f is the indicator function, we get the Lagrangian relaxation and the previous saddle point formulation of LP.

Different penalty functions

What if we use a different penalty function f for the primal constraint, such as:

$$f(Ax) = e^{|Ax-b|}$$

What we get as **convex conjugate** function?

Research Question 1: Different Saddle-point formulations

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + \langle c, x \rangle - \langle y, b \rangle$$

If f is the indicator function, we get the Lagrangian relaxation and the previous saddle point formulation of LP.

Different penalty functions

What if we use a different penalty function f for the primal constraint, such as:

$$f(Ax) = e^{|Ax-b|}$$

What we get as **convex conjugate** function? Can we solve efficiently the proximal operator?

Research Question 1: Different Saddle-point formulations

Consider the problem

$$\min_{x \in X} f(Kx) + g(x)$$

where f, g are convex, l.s.c. functions and $K : X \rightarrow Y$ is a bounded linear operator.

Then, we can write the *saddle point* problem

$$\min_{x \in X} \max_{y \in Y} \langle y, Kx \rangle + \langle c, x \rangle - \langle y, b \rangle$$

If f is the indicator function, we get the Lagrangian relaxation and the previous saddle point formulation of LP.

Different penalty functions

What if we use a different penalty function f for the primal constraint, such as:

$$f(Ax) = e^{|Ax-b|}$$

What we get as **convex conjugate** function? Can we solve efficiently the proximal operator?

Interesting functions to look at: [Energy](#), [Shannon Entropy](#), [Fermi-Dirac](#), [Hellinger](#), and [Burg](#) (e.g., see (Teb18)).

Research Question 2: Using a deflection techniques

In Lagrangian relaxation approach for Integer Linear Programming problems a simple but effective techniques is the **deflection** techniques, which mitigate the zig-zag effect of subgradient methods.

We can “rephrase” this technique to modify the *over relaxation* step in PDLP as follows:

$$\begin{aligned}x^{k+2} &= \text{prox}_{\tau g}(x^{k+1} - \tau A^\top y^{k+1}) \\y^{k+2} &= \text{prox}_{\sigma f^*}(y^{k+1} + \sigma A(3x^{k+2} - x^{k+1} - x^k)).\end{aligned}$$

Research Question 2: Using a deflection techniques

In Lagrangian relaxation approach for Integer Linear Programming problems a simple but effective techniques is the **deflection** techniques, which mitigate the zig-zag effect of subgradient methods.

We can “rephrase” this technique to modify the *over relaxation* step in PDLP as follows:

$$\begin{aligned}x^{k+2} &= \text{prox}_{\tau g}(x^{k+1} - \tau A^\top y^{k+1}) \\y^{k+2} &= \text{prox}_{\sigma f^*}(y^{k+1} + \sigma A(3x^{k+2} - x^{k+1} - x^k)).\end{aligned}$$

QUESTION 1: Does it works at least computationally?

Research Question 2: Using a deflection techniques

In Lagrangian relaxation approach for Integer Linear Programming problems a simple but effective techniques is the **deflection** techniques, which mitigate the zig-zag effect of subgradient methods.

We can “rephrase” this technique to modify the *over relaxation* step in PDLP as follows:

$$\begin{aligned}x^{k+2} &= \text{prox}_{\tau g}(x^{k+1} - \tau A^\top y^{k+1}) \\y^{k+2} &= \text{prox}_{\sigma f^*}(y^{k+1} + \sigma A(3x^{k+2} - x^{k+1} - x^k)).\end{aligned}$$

QUESTION 1: Does it works at least computationally?

OPEN QUESTION 2: Does it converge to a saddle point?

Research Question 2: Using a deflection techniques

In Lagrangian relaxation approach for Integer Linear Programming problems a simple but effective techniques is the **deflection** techniques, which mitigate the zig-zag effect of subgradient methods.

We can “rephrase” this technique to modify the *over relaxation* step in PDLP as follows:

$$\begin{aligned}x^{k+2} &= \text{prox}_{\tau g}(x^{k+1} - \tau A^T y^{k+1}) \\y^{k+2} &= \text{prox}_{\sigma f^*}(y^{k+1} + \sigma A(3x^{k+2} - x^{k+1} - x^k)).\end{aligned}$$

QUESTION 1: Does it works at least computationally?

OPEN QUESTION 2: Does it converge to a saddle point?

OPEN QUESTION 3: If yes, which is the converge rate?

Take away message

To deeply understand a computational paper you have to try to reproduce the numerical results,

Take away message

To deeply understand a computational paper you have to try to reproduce the numerical results, since you are smart enough to find an **incremental** but **solid** improvement.

Take away message

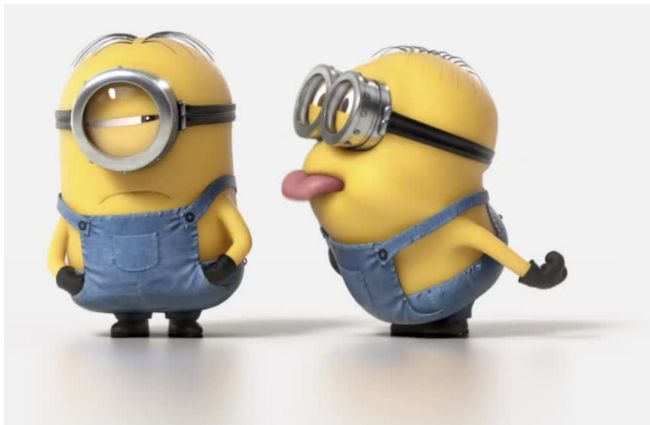
To deeply understand a computational paper you have to try to reproduce the numerical results, since you are smart enough to find an **incremental** but **solid** improvement.

Questions?



References for these slides

- *Missing in action ...*



References I

- [AHLL21] David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *arXiv preprint arXiv:2105.12715*, pages 1–20, 2021.
- [CP11] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.
- [CP16] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.
- [PC11] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, pages 1762–1769. IEEE, 2011.
- [Roc76] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.
- [Teb18] Marc Teboulle. A simplified view of first order methods for optimization. *Mathematical Programming*, 170(1):67–96, 2018.