Project Non-ML 5
Final Project for the CM course 2020/21

Matteo De Francesco

Contents

1 Introduction

In this report we will analize the convex quadratic problem

$$\min \left\{ x^\intercal Q x + q^\intercal x : \sum_{i \in I^k} x_i = 1, k \in K, x \geq 0 \right\} \tag{P}$$

with the following constraints: $x \in \mathbb{R}^n$, the index sets I^k form a partition of the set $\{1, \ldots, n\}$ (i.e.$\cup_{k \in K} I^k = \{1, \ldots, n\}$, and $I^h \cap I^k = \emptyset$ for all h and k), and Q is positive semidefinite.
The aim of this project is to exploit the Lagrangian dual problem and solve it via one of the *deflected subgradient* methods.
We can identify the inequality and equality constraints and rewrite them in the typical form

$$G(x) \to g_i(x) \leq 0 \implies -x_i \leq 0 \; \forall i \in \{1, \ldots, n\}$$
$$H(x) \to h_j(x) = 0 \implies \sum_{i \in I^k} x_i - 1 = 0 \; \forall k \in K$$

Hence we can rewrite the problem in the following form:

$$\begin{cases} \min \; x^\intercal Q x + q^\intercal x \\ -x_i \leq 0 \; \forall \, i \in \{1, \ldots, n\} \\ \sum_{i \in I^k} x_i - 1 = 0 \; \forall \, k \in K \end{cases}$$

In order to have less dual variables, we can rewrite the above problem as a lagrangian function of only the inequality constraints:

$$\mathcal{L}(x; \lambda) = x^\intercal Q x + q^\intercal x - \langle \lambda, x \rangle$$

Now, we can easily construct the lagrangian dual function considering the equality constraints

$$\psi(\lambda) = \min_{x \in \mathcal{Y}} \{ \mathcal{L}(x; \lambda), \; \lambda \geq 0 \}$$

where $\mathcal{Y} = \{ \sum_{i \in I^k} x_i = 1 \; \forall \, k \in K \}$. Hence the following optimization problem

$$\begin{cases} \max_\lambda \; \psi(\lambda) \\ \text{subject to } \lambda \geq 0 \end{cases} \tag{D}$$

We are assuming that optimizing over the set $\mathcal{Y}$ can be done very easily.
In the next section, we will briefly recall the properties of the `ADAGRAD` family of algorithms[1].

2 Deflected subgradient Analysis

2.1 `ADAGRAD` introduction

The projection rule of a point y onto the constraint set $\mathcal{X}$ according to a positive semidefinite matrix A amounts to:

$$\prod_{\mathbb{X}}^{A}(y) = \arg\min_{\lambda \in X} \|\lambda - y\|_A = \arg\min_{\lambda \in X} \langle \lambda - y, A(\lambda - y) \rangle$$

and it's aimed to minimize the Mahalanobis norm.
`ADAGRAD` applies the following projection rule:

$$\lambda_{t+1} = \prod_{\mathcal{X}}^{diag(G_t)^{1/2}} (\lambda_t - \eta \, diag(G_t)^{-1/2} g_t) \tag{1}$$

where the matrix A coincides with the diagonal square root of the full outer product of the subgradients $G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^\intercal$. As we stated in the introduction, we should exploit `ADAGRAD`

on the dual problem, whose constraint set consists in the simple inequality constraint $\lambda \geq 0$, which is addressable by `ADAGRAD`, given the fact that the algorithm can be applied to any convex set $\mathcal{X} \subseteq \mathbb{R}^n$, respected by our problem (we are in $\mathbb{R}^n_+$).
In addition, as the general projection rule tell us, the new update λ_{t+1} is projected over the constraint set $\mathcal{X}$ according to the matrix of the gradients.
We will recall in the next section some algorithmic properties of the algorithm convergence for diagonal matrices, and we will derive the update rule followed by the projection over the constraint set $\mathcal{X} = \{\lambda \geq 0\}$.

2.2 Algorithm characteristics

First of all, we reiterate that `ADAGRAD` is suitable for the dual problem, given its application in a convex setting. The goal is to attain a small regret bound:

$$R_\phi(T) \triangleq \sum_{t=1}^{T} \psi(\lambda_t) - \psi(\lambda^*)$$

between the actual value of the dual function ψ with the corresponding iterate λ at step t and the value of ψ with the optimal solution λ^*.
The point of `ADAGRAD` is that not all features are equal, hence they must be treated differently. That's the purpose of using an adaption to the geometry of space, so do not use anymore a standard gradient descent but conditioning the different values based on a positive semidefinite matrix A (the G_t in this case).
Two algorithm versions are analyzed in [1], where we omitted the regularization term due to our problem setting. The first update, referred to as *primal-dual subgradient method*, is

$$\lambda_{t+1} = \arg\min_{\lambda \in \mathcal{X}} \{\eta \langle \overline{g}_t, \lambda \rangle + \frac{1}{t} \Psi_t(\lambda)\} \tag{2}$$

coming from [2], where $\overline{g}_t = \frac{1}{t} \sum_{\tau=1}^{t} g_\tau$ is the average gradient, η is a fixed stepsize and Ψ_t is the *proximal term*.
The second update instead

$$\lambda_{t+1} = \arg\min_{\lambda \in \mathcal{X}} \{\eta \langle g_t, \lambda \rangle + B_{\Psi_t}(\lambda, \lambda_t)\} \tag{3}$$

where B refers to the Bregman divergence. This second update comes from [3].
Finally, also the previous mentioned projection rule can be used as an update:

$$\lambda_{t+1} = P_{\mathcal{X}}\{\lambda_t + \eta \, diag(G_t)^{-1/2} g_t\} \tag{4}$$

where P denotes the projection operation. The proximal term Ψ_t is the key point of the algorithm. Instead of using a fixed proximal functions, both the updates use squared Mahalanobis norms as their proximal functions, setting then $\Psi_t(\lambda) = \langle \lambda, H_t \lambda \rangle$ for a symmetric matrix $H_t \succeq 0$. In particular, the diagonal case which we will recall here make use of:

$$H_t = \delta I + diag(G_t)^{1/2}$$

for some smalled fixed $\delta \geq 0$.
The usage of a strongly convex proximal function is also remarked in [4] in appendix A.

2.3 Proximal term discussion

In order to attain a lower regret bound and to adapt to the geometry of the space, the objective of the authors is to not use anymore a standard proximal functions but a modified version of it.
The proximal function act as a regularization term, typically the Euclidian projection over a convex set. Given the indicator function of a convex set C:

$$I_C(\lambda) = \begin{cases} 0 & \lambda \in C \\ +\infty & \text{otherwise} \end{cases}$$

the standard proximal mapping of I_C is the Euclidean projection on C:

$$prox_{I_C}(\lambda) = \arg\min_{u \in C} \|u - \lambda\|_2^2 = P_C(\lambda)$$

Instead in this case the authors noticed that some local region of the function to be optimized need more "attention than others". What they come up with then is the modification (also remarked in the `ADAGRAD` introduction) using a different proximal function where

$$prox_{I_C}(\lambda) = \arg\min_{u \in C} \|u - \lambda\|_A^2 = P_C(\lambda)$$

the Euclidean projection is not computed anymore according to a 2-norm, but according to a matrix A which in the case of `ADAGRAD` coincide with the matrix of the subgradients. A summary of the properties and aspects of some proximal functions can be found in [5]. Below is reported the analysis of the proximal function regarding [1].
Examining the regret bounds for the updates in [6] and [2], it is quite obvious that they depends on dual norms of the derivative of the function to be optimized, and in turn this last depend on the choice of Ψ. The objective of [1] is to properly modify the value of Ψ during the run of the algorithm in order to lower the contribution of the norms, and so lower the regret bound. This is achieved by keeping second order information about the sequence of iterates λ_t and allow Ψ to vary on each round of the algorithm. To achieve this, we must assume that Ψ_t is monotically non-decreasing, 1-strongly convex with respect to a time-dependent semi-norm $\|\cdot\|_{\Psi_t}$. Formally, given two generic points x and y, Ψ is 1-strongly convex with respect to $\|\cdot\|_\Psi$ if

$$\Psi(y) \geq \Psi(x) + \langle \nabla\Psi(x), y - x \rangle + \frac{1}{2}\|x - y\|_\Psi^2$$

As a consequence, strong convexity is guaranteed if and only of $B_{\Psi_t}(x, y) \geq \frac{1}{2}\|x - y\|_{\Psi_t}^2$. The following bound holds then on proximal term, respectively for (2) and (3). Proofs can be found in Appendix F of [1].

Proposition 2.1. *Let the sequence $\{\lambda_t\}$ be defined by the update (2). For any $\lambda^* \in \mathcal{X}$*

$$\sum_{t=1}^{T} \psi(\lambda_t) - \psi(\lambda^*) \leq \frac{1}{\eta}\Psi_T(\lambda^*) + \frac{\eta}{2}\sum_{t=1}^{T} \|\psi'(\lambda_t)\|_{\Psi_{t-1}^*}^2$$

Proposition 2.2. *Let the sequence $\{\lambda_t\}$ be defined by the update (3). For any $\lambda^* \in \mathcal{X}$*

$$\sum_{t=1}^{T} \psi(\lambda_t) - \psi(\lambda^*) \leq \frac{1}{\eta}B_{\Psi_1}(\lambda^*, \lambda_1) + \frac{1}{\eta}\sum_{t=1}^{T-1}\left[B_{\Psi_{t+1}}(\lambda^*, \lambda_{t+1}) - B_{\Psi_t}(\lambda^*, \lambda_{t+1})\right] + \frac{\eta}{2}\sum_{t=1}^{T} \|\psi'(\lambda_t)\|_{\Psi_t^*}^2$$

The proximal term and the bregman divergence present in (2) and (3) will be computed according to the subgradient matrix, in order to iteratively modify the update and thus adapting to the geometry of the space.

2.4 Convergence Analysis

First of all, let us state that the convergence rate of `ADAGRAD` is the same of the Stochastic Gradient Descent (`SGD`), hence $O(1/\sqrt{T})$ but with a lower constant due to the use of G matrix.
The algorithm 1 reported in [1] will be applied to our problem and is reported here:

Algorithm 1 `ADAGRAD` for diagonal matrices

function `ADAGRAD`(η,δ)
  $x_1 \leftarrow 0$
  $\lambda_1 \leftarrow 0$
  $g_{1:0} \leftarrow [\,]$
  for $t \leftarrow 1$ to T **do**
    $Loss = f_t(x_t)$
    $g_t \leftarrow \partial\psi(\lambda_{t-1})$ of ψ at λ_{t-1} ▷ Compute subgradient at λ_{t-1}
    $g_{1:t} \leftarrow [g_{1:t-1}\ g_t]$ ▷ Store subgradient
    $s_{t,i} \leftarrow \|g_{1:t,i}\|_2$ ▷ Compute the optimal $s_{t,i}$
    $H_t \leftarrow \delta I + diag(s_t)$
    $\Psi_t(\lambda) \leftarrow \frac{1}{2}\langle \lambda, H_t\lambda\rangle$
    Either compute (2) or (3)
  end for
end function

The general convergence result of this algorithm for both the updates is reported in theorem 5 of the original paper. We will now report the theoretical analysis behind procedure 1 and lastly modify the algorithm in order to match our problem settings.

Theorem 2.1. *Let the sequence $\{\lambda_t\}$ be defined by algorithm 1. For λ_t generated using the update* (2) *with $\delta \geq \max_t \|g_t\|_\infty$, for any $\lambda^* \in \mathcal{X}$*

$$R_\phi(T) \leq \underbrace{\frac{\delta}{\eta}\|\lambda^*\|_2^2 + \frac{1}{\eta}\|\lambda^*\|_\infty^2 \sum_{i=1}^{d}\|g_{1:T,i}\|_2}_{①} + \underbrace{\eta\sum_{i=1}^{d}\|g_{1:T,i}\|_2}_{②} \tag{5}$$

For λ_t generated using the update (3)*, for any $\lambda^* \in \mathcal{X}$*

$$R_\phi(T) \leq \underbrace{\frac{1}{2\eta}\max_{t\leq T}\|\lambda^* - \lambda_t\|_\infty^2 \sum_{i=1}^{d}\|g_{1:T,i}\|_2}_{③} + \underbrace{\eta\sum_{i=1}^{d}\|g_{1:T,i}\|_2}_{②} \tag{6}$$

We will now delve into analyzing the different $\textcircled{\cdot}$ parts.
First of all, we should recall some intuition in order to understand better the regret bound provided above.
Focusin on the algorithm 1, the chosen value of $s_{t,i}$ explain us why the particular choice of the proximal function $\Psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$ is so important. $s_{t,i}$ comes from the solution of the problem

$$\min_s \sum_{t=1}^{T}\sum_{i=1}^{d}\frac{g_{t,i}^2}{s_i} \text{ s.t. } s \succeq 0, \langle 1, s\rangle \leq c$$

solved by optimizing the Lagrangian

$$\mathcal{L}(s,\lambda,\theta) = \sum_{i=1}^{d}\frac{\|g_{1:T,i}\|_2^2}{s_i} - \langle\lambda, s\rangle + \theta(\langle 1, s\rangle - c)$$

Taking the partial derivatives to find the infimum of $\mathcal{L}$, and using the complementary slackness condition on $\lambda_i s_i$ imply that $\lambda_i = 0$. As a consequence, we obtain $s_i = c\|g_{1:T,i}\|_2 / \sum_{j=1}^{d}\|g_{1:T,j}\|_2$. Plugging this into the previous objective function, we get

$$\inf_s\left\{\sum_{t=1}^{T}\sum_{i=1}^{d}\frac{g_{t,i}^2}{s_i} : s \succeq 0, \langle 1, s\rangle \leq c\right\} = \frac{1}{c}\left(\sum_{i=1}^{d}\|g_{1:T,i}\|_2\right)^2$$

Now it is natural to suspect that for s achieving the infimum in this latter equation, using a proximal function similar to $\Psi(\lambda) = \langle\lambda, diag(s)\lambda\rangle$ with associated squared dual norm

$\|\lambda\|^2_{\Psi^*} = \langle \lambda, diag(s)^{-1}\lambda \rangle$ we should lower the gradient terms both in (5) and (6).
The upper bound on the gradient term for both the updates is taken from the `LEMMA 4` of [1], stating:

Lemma 2.1. *Let $g_t = \psi'(\lambda_t)$ and $g_{1:t}$ and s_t be defined as in algorithm 1. Then*

$$\sum_{t=1}^{T} \langle g_t, diag(s_t)^{-1} g_t \rangle \leq 2 \sum_{i=1}^{d} \|g_{1:T,i}\|_2$$

To obtain a bound, we need to consider the terms consisting of the dual-norm of the subgradient in the bounds (5) and (6), which is $\|\psi'(\lambda_t)\|^2_{\Psi_t^*}$. When we choose $\Psi_t(\lambda) = \langle \lambda, (\delta I + diag(s_t))\lambda \rangle$, the associated dual norm is

$$\|g\|^2_{\Psi_t^*} = \langle g, (\delta I + diag(s_t))^{-1} g \rangle$$

Following from the definition of s_t in 1, we have $\|\psi'(\lambda_t)\|^2_{\Psi_t^*} \leq \langle g_t, diag(s_t)^{-1} g_t \rangle$. Thus we have the following implication

$$\sum_{t=1}^{T} \|\psi'(\lambda_t)\|^2_{\Psi_t^*} \leq \sum_{i=1}^{d} \|g_{1:T,i}\|_2$$

which prove the regret term (2).
Consequently, it remains to prove the bound over the Bregman divergence and the term $\Psi_T(\lambda^*)$ of proposition 2.1 and 2.2. Focusing on the composite mirror descent, we have:

$$\begin{aligned} B_{\Psi_{t+1}}(\lambda^*, \lambda_{t+1}) - B_{\Psi_t}(\lambda^*, \lambda_{t+1}) &= \frac{1}{2} \langle \lambda^* - \lambda_{t+1}, diag(s_{t+1} - s_t)(\lambda^* - \lambda_{t+1}) \rangle \\ &\leq \frac{1}{2} \max_i (\lambda_i^* - \lambda_{t+1,i})^2 \|s_{t+1} - s_t\|_1 \end{aligned}$$

Since $\|s_{t+1} - s_t\|_1 = \langle s_{t+1} - s_t, 1 \rangle$ and $\langle s_T, 1 \rangle = \sum_{i=1}^{d} \|g_{1:T,i}\|_2$ we have

$$\begin{aligned} \sum_{t=1}^{T-1} B_{\Psi_{t+1}}(\lambda^*, \lambda_{t+1}) - B_{\Psi_t}(\lambda^*, \lambda_{t+1}) &\leq \frac{1}{2} \sum_{t=1}^{T-1} \|\lambda^* - \lambda_{t+1}\|^2_\infty \langle s_{t+1} - s_t, 1 \rangle \\ &\leq \frac{1}{2} \max_{t \leq T} \|\lambda^* - \lambda_t\|^2_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2 - \frac{1}{2} \|\lambda^* - \lambda_1\|^2_\infty \langle s_1, 1 \rangle \end{aligned}$$

which prove us the term (3).
Finally, we also have that

$$\Psi_T(\lambda^*) = \delta \|\lambda^*\|_2^2 + \langle \lambda^*, diag(s_T)\lambda^* \rangle \leq \delta \|\lambda^*\|_2^2 + \|\lambda^*\|^2_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2$$

which give us the bound (1).
Performing a few algebraic simplification lead us to a corollary which give us a more intuitive form of the regret bound. Assume that $\mathcal{X}$ is a compact set and set $D_\infty = \sup_{\lambda \in \mathcal{X}} \|\lambda - \lambda^*\|_\infty$, and define

$$\gamma_T \triangleq \sum_{i=1}^{d} \|g_{1:T,i}\|_2 = \inf_s \left\{ \sum_{t=1}^{T} \langle g_t, diag(s)^{-1} g_t \rangle : \langle 1, s \rangle \leq \sum_{i=1}^{d} \|g_{1:T,i}\|_2, s \succeq 0 \right\}$$

Corollary 2.1. *Assume that D_∞ and γ_T are defined as above. For $\{\lambda_t\}$ generated using the (2) with $\eta = \|\lambda^*\|_\infty$, for any $\lambda^* \in \mathcal{X}$ we have*

$$R_\phi(T) \leq 2\|\lambda^*\|_\infty \gamma_T + \delta \frac{\|\lambda^*\|_2^2}{\|\lambda^*\|_\infty} \leq 2\|\lambda^*\|_\infty \gamma_T + \delta \|\lambda^*\|_1$$

Using update (3) *to generate* $\{\lambda_t\}$ *and setting* $\eta = D_\infty/\sqrt{2}$*, we have*

$$R_\phi(T) \leq \sqrt{2} D_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2 = \sqrt{2} D_\infty \gamma_T$$

All these presented results are respected by our problem. Indeed, as we stated before, our original problem is for sure a convex problem (a quadratic function with positive semidefinite Q) and also the constraints are convex (a set of disjoint unitary simplices) implying strong duality. As we know from theory, the dual problem is for sure convex, even if the original one it's not. What we should argue is the presence of the constraints on the dual variables. Being `ADAGRAD` applicable to any convex set $\mathcal{X} \subseteq \mathbb{R}^n$, this is suitable for our problem, since $\mathcal{X} = \{\lambda \geq 0\}$. We will use one of the just explained update, (preferably the *primal-dual* one, since we have a general proof of convergence over exactly $\mathbb{R}^n_+$, in `LEMMA 2` of Appendix A of [4]), compute the update and then project it on the λ's constraint set.
We expect to achieve an asymptotically sub-linear regret, as the authors showed in their work.

2.5 General application to our case

In our setting, we aim to solve problem (P) using (D). We will use $\mathcal{X} = \{\lambda \geq 0\}$ as the constraint set of the lagrangian multipliers λ's, and $\mathcal{Y} = \{\sum_{i \in I^k} x_i = 1 \ \forall k \in K\}$ as the constraint set of the primal variables.
To solve the problem (2) and (3) in our problem settings, we consider the update of the dual variable λ. Referring to the problem (D), we can freely choose among three different update rules, in order (1), (2) and (3):

$$\begin{aligned} \lambda_{t+1} &= P_{\mathcal{X}}\{\lambda_t + \eta \, diag(G_t)^{-1/2} g_t\} \\ \lambda_{t+1} &= \arg\max_{\lambda \in \mathcal{X}} \{\eta \langle \overline{g}_t, \lambda \rangle + \frac{1}{t} \Psi_t(\lambda)\} \\ \lambda_{t+1} &= \arg\max_{\lambda \in \mathcal{X}} \{\eta \langle g_t, \lambda \rangle + B_{\Psi_t}(\lambda, \lambda_t)\} \end{aligned} \tag{7}$$

We need to find a maximum for the considered update function and then project it onto the constraint set $\mathcal{X}$.
About the terms Ψ and B_Ψ, these will be replaced, according to what we have said before, respectively by:

$$\begin{aligned} \Psi_t(\lambda) &= \frac{1}{2} \langle \lambda, H_t \lambda \rangle \\ B_{\Psi_t}(\lambda, \lambda_t) &= \Psi_t(\lambda) - \Psi_t(\lambda_t) - \langle \nabla \Psi_t(\lambda_t), \lambda - \lambda_t \rangle \end{aligned}$$

Regarding the maximum, this can be simply solved by differentiating the function with respect to the variable to be maximized, and setting it equal to 0. The detailed derivation of each update can be found in the appendix A.

$$\hat{\lambda}_{t+1} = \lambda_t + \eta diag(G_t)^{-1/2} g_t \qquad \text{(Update (4))}$$

$$\hat{\lambda}_{t+1} = -H_t^{-1} t \, \eta \, \overline{g}_t \qquad \text{(Update (2))}$$

$$\hat{\lambda}_{t+1} = \lambda_t - \eta \, H_t^{-1} \, g_t \qquad \text{(Update (3))}$$

For the stepsize η, this can be fixed apriori and we will see being it crucial for the convergence of the algorithm. Different stepsize rules can be employed [7]

$$\eta = h \qquad \text{Constant step size rule, with } h > 0 \tag{9}$$

$$\eta = \frac{h}{\|g^t\|} \qquad \text{Constant step length} \tag{10}$$

$$\eta = \frac{\alpha}{\beta + t} \qquad \text{Square summable but not summable, with } \alpha > 0 \text{ and } \beta \geq 0 \tag{11}$$

$$\eta = \frac{\alpha}{\sqrt{t}} \qquad \text{Nonsummable diminishing} \tag{12}$$

$$\eta = \frac{f(x^*) - \phi(\lambda_t)}{\|g^t\|^2} \qquad \text{Polyak stepsize} \tag{13}$$

For the dual variables, after we obtain $\hat{\lambda}_{t+1}$, we use the projection over the nonnegative orthant to get λ_{t+1}, formally

$$\lambda_{t+1} = P_{\mathcal{X}}(\hat{\lambda}_{t+1}) = \max\{0, \hat{\lambda}_{t+1}\}$$

which is a trivial problem tackled many times in the literature [8].
We need also to derive an update for the primal variables, which are needed at each iteration in order to compute the dual function $\psi(\lambda)$ and the subgradient $\nabla_\lambda \psi(\lambda)$. Given the previous multiplier value λ_t

$$x_{t+1} = \arg\min_{x \in \mathcal{Y}} \{x^\intercal Q x + q^\intercal x - \langle \lambda_t, x \rangle\}$$

We need to solve the lagrangian relaxation of a convex quadratic problem, depending on equality constraints made up of disjoint simplices. The constraint set $\mathcal{Y}$ can be rewritten in a matrix form $Ax = b$, where the vector b is a $k \times 1$ vector of all ones, and the matrix A can be derived with the following simple algorithm:

Algorithm 2 Construct matrix A

```
procedure CONSTRUCT_A(K, [I^k])
    A ← [ ]                                  ▷ Initialize k × n empty matrix
    for k ← 1 to K do
        a_k ← zeros(n, 1)                    ▷ n × 1 empty row vector
        for i ← 1 to n do
            if i ∈ I^k then                  ▷ Check if index i is in the set I^k
                a_k[i] ← 1
            end if
        end for
        A[k, :] ← a_k
    end for
end procedure
```

Using the **KKT** (Karush-Kuhn Tucker) conditions, we can solve the problem directly through linear algebra by constructing the following linear system:

$$\begin{cases} 2Qx + q - \lambda_t + \mu A^\intercal = 0 \\ Ax - b = 0 \end{cases} \implies \begin{cases} 2Qx + \mu A^\intercal = \lambda_t - q \\ Ax = b \end{cases} \implies$$

$$\begin{bmatrix} 2Q & A^\intercal \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \mu \end{bmatrix} = \begin{bmatrix} \lambda_t - q \\ b \end{bmatrix}$$

a linear system that can be solved directly and efficiently by computing the pivoted $LDL^\intercal$ factorization of the matrix, given the fact that we have a symmetric positive semidefinite matrix. We used the `Bunch-Kaufman` factorization [9], a variant of the $LDL^\intercal$, which use partial pivoting, it is numerically stable and twice as fast as the LU factorization. Given the `Bunch-Kaufman` factorization of A:

$$PAP^\intercal = LDL^\intercal$$

We can exploit the linear system:

$$\begin{bmatrix} 2Q & A^\intercal \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \mu \end{bmatrix} = \begin{bmatrix} \lambda_t - q \\ b \end{bmatrix} \implies \begin{bmatrix} x \\ \mu \end{bmatrix} = \begin{bmatrix} 2Q & A^\intercal \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} \lambda_t - q \\ b \end{bmatrix} \implies$$

$$\begin{bmatrix} x \\ \mu \end{bmatrix} = P \left(L^{-\intercal} \left(D^{-1} \left(L^{-1} \left(P^\intercal \begin{bmatrix} \lambda_t - q \\ b \end{bmatrix} \right) \right) \right) \right)$$

which can be solved efficiently using matrix vector products and forward substitution (for the matrix L) with a complexity of $\approx O((n+K)^2)$.
Lastly, regarding the stopping condition for the algorithm termination, we can check the dual variables residual. Follows that, given a certain value of tolerance ε, we have

$$\|\lambda_t - \lambda_{t-1}\|_2 \leq \varepsilon \tag{14}$$

In addition to this, what we can check too is the duality gap between the original function and the dual function (provided that we know the optimal value), so

$$f(x) - \psi(\lambda)$$
$$f(x) - f(x^*) \leq f(x) - \psi(\lambda)$$

From this we understand that having a zero duality gap implies optimality of the solution, and so fixing a certain value of tolerance τ we have a second stopping condition, computing the relative gap error

$$\frac{f(x^*) - \psi(\lambda_i)}{|f(x^*)|} \leq \tau \tag{15}$$

for each λ_i computed during iterations.
The starting value of the λ's iterates is a vector of ones and the starting x is computed immediately using the starting λ and the factorization explained before.
Putting everything together, we obtain a slightly different algorithm than 1:

Algorithm 3 `ADAGRAD` on our dual problem

function `ADAGRAD`(η, δ, ε, τ, max_iter, $defl$)
  $\lambda_0 \leftarrow 1$
  $g_{1:0} \leftarrow [\,]$
  $x_0 \leftarrow$ system solution
  $\psi = \psi(\lambda_0)$
  for $t \leftarrow 1$ to max_iter **do**
    $g_t \leftarrow \dfrac{\partial \psi_\lambda(\lambda)}{\partial \lambda}$ ▷ Compute subgradient B
    if $defl$ **then** ▷ Compute deflection
      Fix γ
      Update $d_t = \gamma g_t + (1-\gamma) d_{t-1}$
    end if
    $g_{1:t} \leftarrow [g_{1:t-1}\ g_t]$ ▷ Store subgradient
    $s_{t,i} \leftarrow \|g_{1:t,i}\|_2$ ▷ Solution of the problem 2.4
    $H_t = \leftarrow \delta I + diag(s_t)$
    $x_t = \arg\min_{x \in \mathcal{Y}} \{x^\intercal Q x + q^\intercal x - \langle \lambda_{t-1}, x \rangle\}$ ▷ Lagrangian
    if $\psi(\lambda_{t-1}) \geq \psi$ **then**
      $\psi = \psi(\lambda_{t-1})$
    end if
    $\hat{\lambda}_t =$ one among (7)
    $\lambda_t = P_{\mathcal{X}}(\hat{\lambda}_t)$ ▷ If $defl$ use d_t where needed
    if (14) `OR` (15) is *true* **then**
      return ψ
    end if
  end for
end function

Since we are using iterative methods based on the standard subgradient method, the number of iterations required in the worst case to reach the objective is given by $\Theta(1/\tau^2)$, where τ is the dual gap we wish to obtain.

3 Implementation

The ideas above described have been implemented using `Julia`. The provided package give us the ability to provide the dimension of the problem n and the number of simplices K. Subsequently, the code is all automated and provide the creation of the matrix $Q \succeq 0$, A, the random sets I^k, the vector q and all the required parameters. The update rules to test can be chosen as well as the different stepsize rules that you want to employ.
The optimal primal solution is found using the off-the-shelf solver function *quadprog()*.
Also, there are some hyperparameters which can be modified by the user inside *main.jl*, like the maximum number of iterations, the ε value to check the λ residual, the tolerance τ as a stopping condition for the dual gap and so on.
At each iteration the customized `ADAGRAD` displays the current iteration t, the time elapsed, the value of $\phi(\lambda_t)$, the change between x_t and x_{t-1} (as a difference in norm), the change of λ_t and λ_{t-1}, the dual gap $f(x^*) - \phi(\lambda_t)/|f(x^*)|$, the norm of the subgradient $\|g_t\|$ and the current stepsize η_t.
All these values are also stored in a *.csv* file, which is used then to generate plots and analyze the results.
In order to test the program on different problem dimensions, we have chosen different sizes to perform our experiments, in order to have a "grid search" of different experimentations:

n	**K**	$\boldsymbol{\varepsilon}$	$\boldsymbol{\tau}$	**max_iter**
1000	$[100, 500]$	10^{-14}	10^{-6}	$[25000, 50000, 100000]$
5000	$[1000, 2500]$			$[10000]$

The code was executed using `Julia REPL` tool. It has a slowish startup time, so the first execution is a "warm-up" execution to compile the entire code.
Q is the resulting covariance matrix of a random generated matrix A_h, in formula:

$$Q = A_h^\intercal \cdot A_h$$

which we know being symmetric and surely positive semidefinite.
Regarding the factorization, the entire `KKT` matrix is factorized using the `Julia` method `factorize()`. The resulting factorization is the `Bunch-Kaufman` factorization, explained before.
Another observation regard the subgradient: being the ψ non differentiable in the general case, the direction is not guaranteed to be ascent/descent. So we keep track of the best $\psi(\lambda)$, best iteration, best x and λ to get the suboptimal solution found.
Last but not least, we noticed in the experiments that the stepsize selection is crucial for the convergence of the algorithm. In particular, the value of the hyperparameters α, β and δ should be chosen ad-hoc to guarantee that we have a good optimization. The mentioned hyperparameters are fundamental, the latter for the update rules (2) and (3) while the first two for the standard (4).
In the first two mentioned updates, the value of H_t^{-1} acts as a sort of "regularizer", scaling in both cases the value of the gradient. Modifying δ impacts primarly on the steps taken.
We can report now the convergence plots of some cases highlighted in table 3. We will show the best results obtained on the three different rules based on the different stepsize available, together with a sum up table.
Also, we will take a look at how the solver scales changing the value of K by means of timing, dimensions and iterations.

4 Results

First result show the comparison between the different update rules using the all the different stepsizes available. We performed experiments using a grid search of different hyperparameters, collecting results and selecting the best ones.
Below is the first result on a problem dimension of $n = 1000$ and $K = 100$:

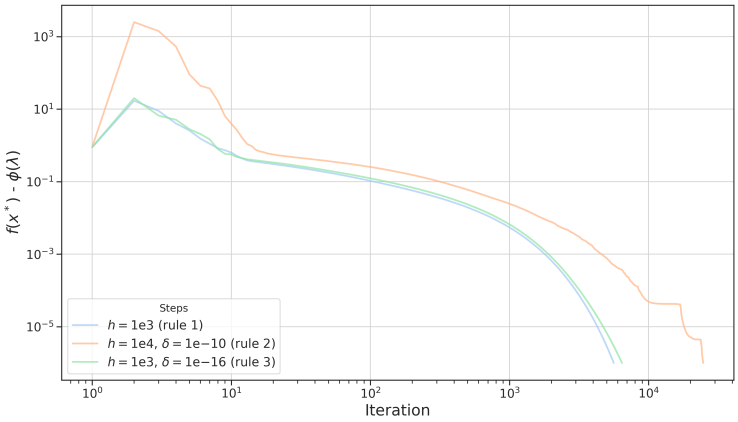

Figure 1: Stepsize (9)

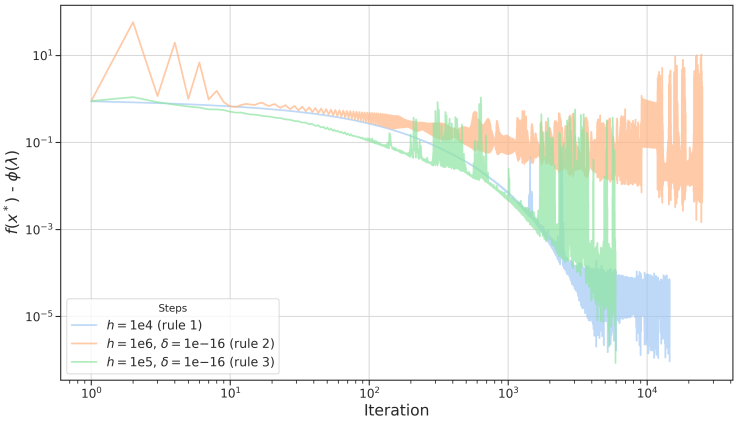

Figure 2: Stepsize (10)

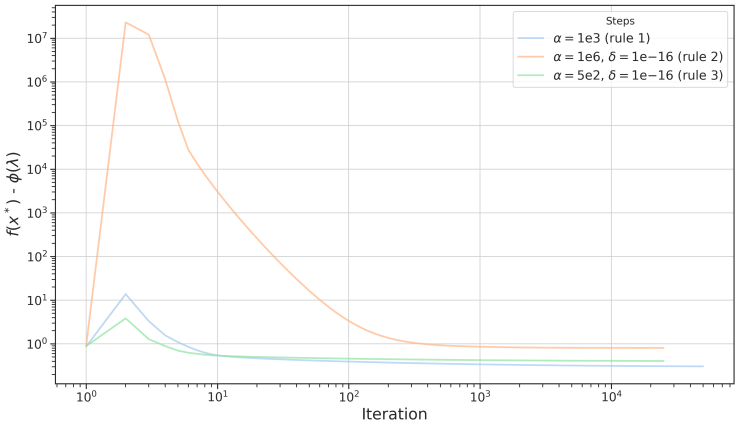

Figure 3: Stepsize (11)

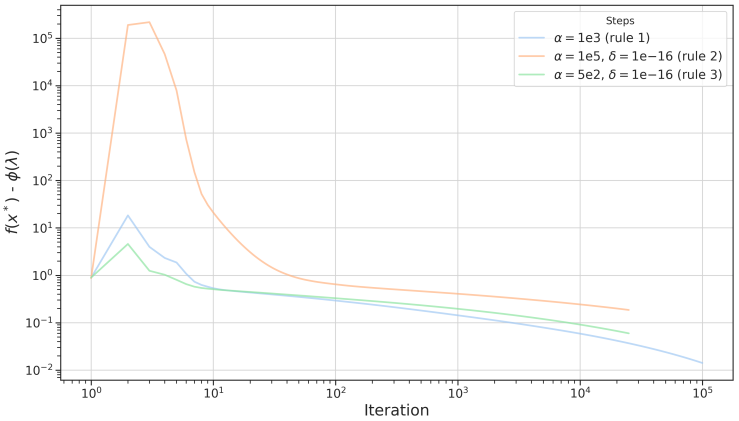

Figure 4: Stepsize (12)

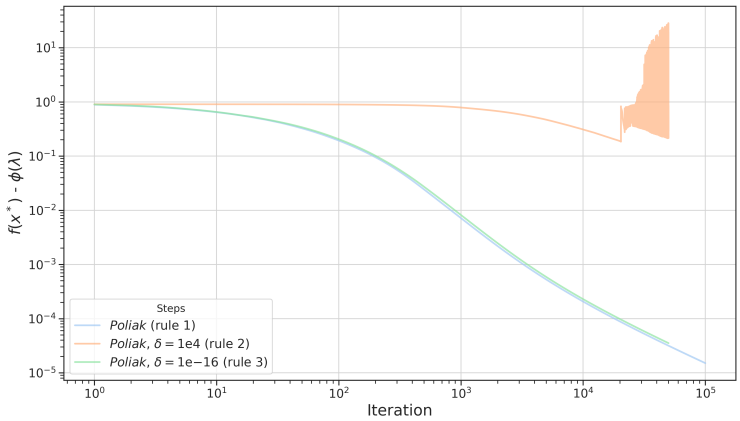

Figure 5: Stepsize (13)

From the above results, we can see how the desired objective of 1e−6 is reached by the first two rules, almost by Poliak stepsize and we are still far with 11 and 12.
A proper selection of the parameter h in figure 1 allow us to reach the objective in a smooth way (almost linear in a doubly logarithmic chart, as expected). Instead in figure 2 we can see how in the tail the gap start oscillating around the objective.

Finally, `Poliak` stepsize (with a proper selection of the hyperparameter δ) follow the same behavior, while results 3 and 4 remain a bit far from the objective. More iterations in these latter case would give a proper convergence.
Here we show a sum up table of the best results from above:

	$h = 1e3$	$h = 1e5$, $\delta = 1e{-}16$	$\alpha = 1e3$	$\alpha = 1e3$	`Poliak`
Stepsize used	(9)	(10)	(11)	(12)	(13)
Update rule	(4)	(3)	(4)	(4)	(4)
Total iterations	5614	5935	50000	100000	100000
total time (sec)	42.1 s	40.2 s	422.2 s	803.2 s	855.5 s
Best $f(x^*) - \psi(\lambda)/\lvert f(x^*)\rvert$	9.995e−7	8.527e−7	3.051e−1	1.416e−2	1.510e−5
Best iteration	5614	5935	50000	100000	100000
Best $\psi(\lambda)$	2.378e6	2.373e6	1.652e6	2.340e6	2.380e6
Best $\lVert\lambda_t - \lambda_{t-1}\rVert$	9.682e−2	3.694e1	2.177e−3	4.941e−2	3.065e−6
Best $\lVert x_t - x_{t-1}\rVert$	1.259e−4	3.652e1	5.122e−5	8.208e−5	2.355e−3
`quadprog` time	9.6 s				
`quadprog` iterations	25				

Table 1: Sum up table for $n = 1000$, $K = 100$

And below a comparison with other tested hyperparameters:

	$h = 1e2$	$h = 1e0$	$\alpha = 1e6$, $\delta = 1e{-}16$	$\alpha = 1e3$
Stepsize used	(9)	(10)	(11)	(12)
Update rule	(4)	(4)	(2)	(3)
Total iterations	6562	5935	25000	25000
total time (sec)	54.7 s	40.2 s	174.7 s	181.5 s
Best $f(x^*) - \psi(\lambda)/\lvert f(x^*)\rvert$	9.998e−7	8.697e−1	8.075e−1	5.973e−2
Best iteration	6562	5935	25000	25000
Best $\psi(\lambda)$	2.365e6	3.096e5	4.565e5	2.224e6
Best $\lVert\lambda_t - \lambda_{t-1}\rVert$	9.059e−2	2.415e−4	1.004e−3	2.007e−1
Best $\lVert x_t - x_{t-1}\rVert$	1.071e−4	1.634e−4	7.051e−5	4.526e−4
`quadprog` time	9.6 s			
`quadprog` iterations	25			

Table 2: Comparison table for $n = 1000$, $K = 100$

To get the best out from our solver, we scaled the problem to see the differences with respect to the previous case. We show now the results for a problem of size $n = 5000$ and $K = 1000$. We omitted the results for the stepsizes (11) and (12), given their slow convergence.

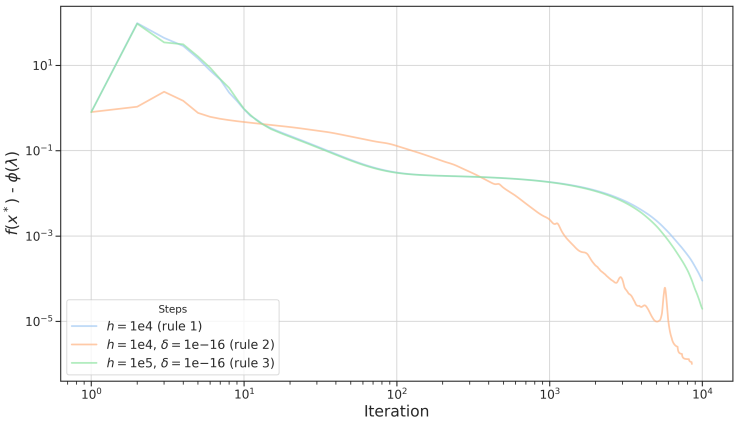

Figure 6: Stepsize (9)

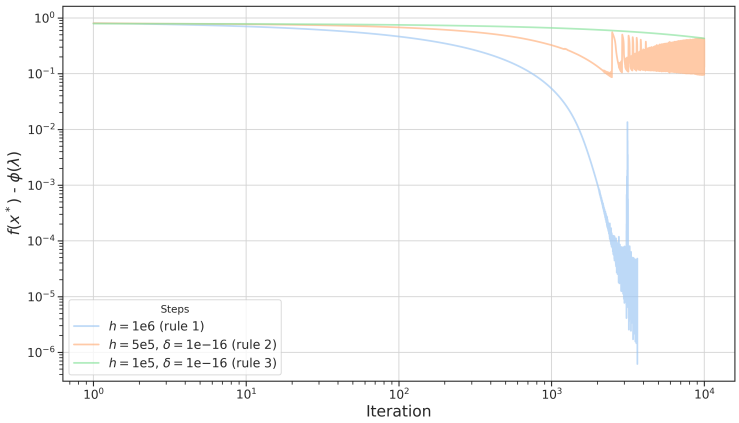

Figure 7: Stepsize (10)

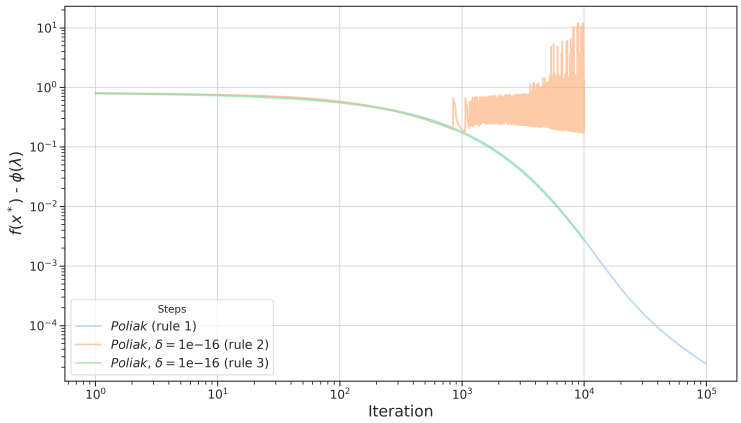

Figure 8: Stepsize (13)

We can see overall that the results are consistent with the previous one. The logarithmic plots show the same trend when reaching the objective, in all the showed results.

	$h = 1e4,\ \delta = 1e{-}16$	$h = 1e6$	`Poliak`
Stepsize used	(9)	(10)	(13)
Update rule	(2)	(4)	(4)
Total iterations	8547	3646	10000
total time (sec)	451.2 s	373.6 s	4445.4 s
Best $f(x^*) - \psi(\lambda)/\lvert f(x^*)\rvert$	9.999e−7	6.179e−7	2.267e−5
Best iteration	8547	3646	100000
Best $\psi(\lambda)$	1.226e9	1.227e9	1.228e9
Best $\lVert\lambda_t - \lambda_{t-1}\rVert$	7.129e1	9.113e2	1.472e−1
Best $\lVert x_t - x_{t-1}\rVert$	6.078e0	6.614e1	4.775e−5
`quadprog` time	117.3 s		
`quadprog` iterations	52		

Table 3: Sum up table for $n = 5000$, $K = 1000$

Finally, for the last example, we scale the number of constraints K to 2500 for a fair comparison

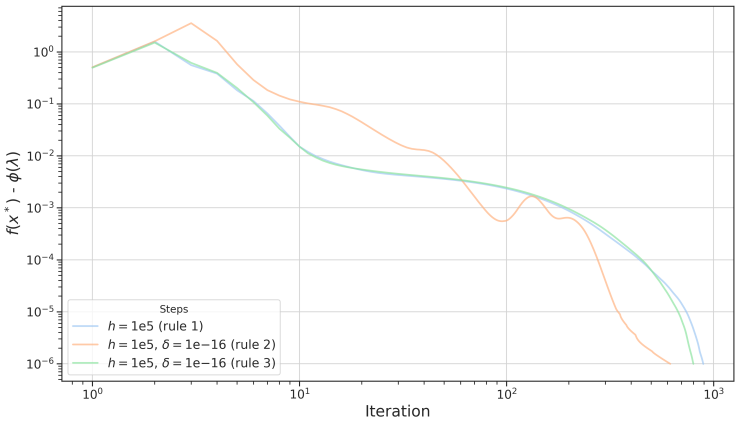

Figure 9: Stepsize (9)

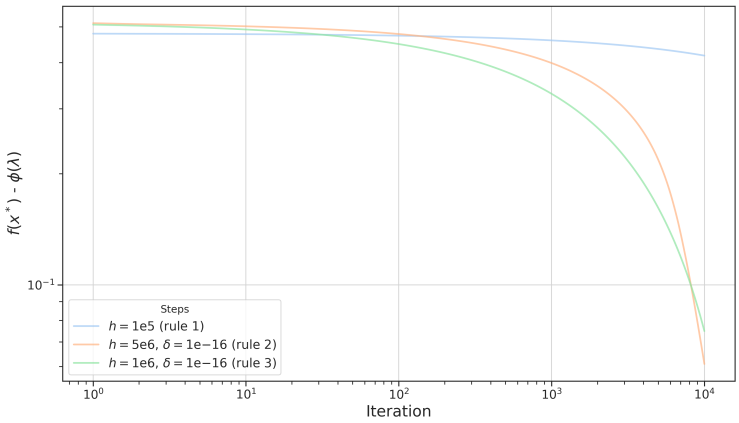

Figure 10: Stepsize (10)

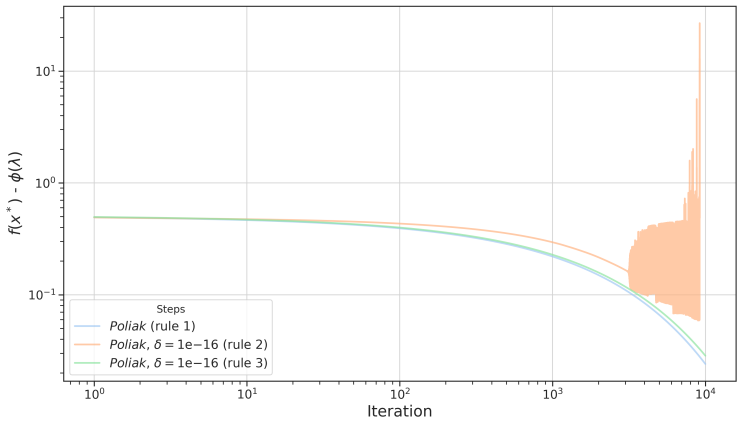

Figure 11: Stepsize (13)

Also here the curves trend is similar to the previous case. Differently from above, in figure 10, we can see how the chosen values of h seems to be more regularized with respect to the previous examples. Here the curve is quite smooth without peaks or obscillation.

	$h = 1e5$, $\delta = 1e{-}16$	$h = 1e6$, $\delta = 1e{-}16$	`Poliak`
Stepsize used	(9)	(10)	(13)
Update rule	(2)	(3)	(4)
Total iterations	618	10000	10000
total time (sec)	43.7 s	691.1 s	709.1 s
Best $f(x^*) - \psi(\lambda)/\lvert f(x^*)\rvert$	9.979e−7	7.493e−2	2.855e−2
Best iteration	618	10000	100000
Best $\psi(\lambda)$	7.756e9	7.172e9	7.537e9
Best $\lVert\lambda_t - \lambda_{t-1}\rVert$	1.982e3	1.47e2	1.001e2
Best $\lVert x_t - x_{t-1}\rVert$	1.922e1	1.179e−1	6.616e−2
`quadprog` time	125.19 s		
`quadprog` iterations	49		

Table 4: Sum up table for $n = 5000$, $K = 2500$

In this last comparison, we show how using an ad-hoc value of h value, we beat the off-the-shelf solver in timing, being 3 times faster.

From the showed examples, and the non showed one we did, we can see the impact that hyperparameters have on convergence. Having a low δ benefits a faster and smooth convergence of (3) and (2), while an high value of this result in a more unstable results. A proper value of h make us converge without peaks around the objective, at least for (9). Stepsize (10) seems to suffer oscillation even adjusting the hyperparameters.
In general, choosing higher values of h and α (than the one found) to speedup the convergence does not give benefits, instead we move far away from the objective. For the α-based stepsize and the `Poliak` one, more iterations should give us a proper convergence. The h-based stepsizes instead are more "brutal", updating a lot the function value at each iteration and so with a bit of luck we reach the objective in a faster way.

5 Code description

The code is provided in `Julia` through package. To try it, simply open the `Julia REPL` inside the current folder, and launch `julia` specifying the option

```
--project=.
```

then enter in the `pkg` mode using `]` and launch

```
instantiate
```

to download the required modules. Then go back to the command line tool (simply backspace) and run

```
include(src/main.jl)
```

At this point you will be asked for a value of n and K.
All the code is compacted into a single script contained inside the `src` folder:

- *main.jl*: containing the required functions;

In addition, the entire project folder contains:

- *papers*: folder containing all the literature referenced in the bibliography;
- *results*: folder containing all the plots and logs obtained with the experimentations values of table 3;
- *util.sh*: a clean-up routine for the *logs* produced;
- *py*: folder containing python script to inspect and plot collected *.csv* files;

References

[1] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[2] Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[3] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *Composite Objective Mirror Descent*, pages 14–26, 12 2010.

[4] Antonio Frangioni, Bernard Gendron, and Enrico Gorgone. On the computational efficiency of subgradient methods: a case study with lagrangian bounds. *Mathematical Programming Computation*, 9(4):573–604, Dec 2017.

[5] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[6] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, Aug 2009.

[7] Marina A. Epelman. Ioe 511/math 652: Continuous optimization methods, section 1, 2007. Lecture notes `http://www-personal.umich.edu/~mepelman/teaching/511notesFA07.pdf`.

[8] Ang Andersen. Projection onto nonnegative orthant, rectangular box and polyhedron, 2020. First draft: March 19, 2020; Last update: December 23, 2020. Université de Mons, `https://angms.science/doc/CVX/Proj_nonnegBoxpoly.pdf`.

[9] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31:163–179, 1975.

A Appendix A: Updates derivations

Given a matrix A of size $n \times n$, the proximal term expression of a vector x of size $n \times 1$ at step t is

$$\Psi_t(x) = \frac{1}{2} \langle x, Ax \rangle$$

A.1 Differentiating proximal term

Derivation of proximal term arises from the need of obtaining the $\arg\max x$ of a given update function. Starting by the definition

$$\frac{\partial \Psi_t(x)}{\partial x} = \frac{1}{2} \frac{\partial}{\partial x} \langle x, Ax \rangle$$

which can be rewritten in a more classical form, and the derivative is immediate

$$\frac{1}{2} \frac{\partial}{\partial x} x^\intercal Ax = Ax$$

A.2 Derivation of primal-dual update

We can now look into the detailed derivation of the update (2). Starting from the equation

$$\lambda_{t+1} = \arg\max_{\lambda \in \mathcal{X}} \{\eta \langle \overline{g}_t, \lambda \rangle + \frac{1}{t} \Psi_t(\lambda)\}$$

we want to achieve the $\arg\max$ over the set $\mathcal{X}$. Focusing on the maximum problem, we want to get the maximum λ, hence:

$$\begin{aligned} \frac{\partial}{\partial \lambda} \eta \langle \overline{g}_t, \lambda \rangle + \frac{\partial}{\partial \lambda} \frac{1}{t} \Psi_t(\lambda) = 0 \\ \eta\, \overline{g}_t + \frac{1}{2t} \frac{\partial}{\partial \lambda} \langle \lambda, H_t \lambda \rangle = 0 \end{aligned}$$

Knowing that H_t is a diagonal matrix and using the previous derivation:

$$\frac{\partial}{\partial \lambda} \langle \lambda, H_t \lambda \rangle = 2\, H_t\, \lambda$$

Substituting this into the above derivation, we get the maximum λ:

$$\begin{aligned} \eta\, \overline{g}_t + \frac{1}{t} H_t\, \lambda = 0 \\ \lambda = -H_t^{-1}\, t\, \eta\, \overline{g}_t \end{aligned}$$

A.3 Derivation of composite-mirror update

We follow the same approach also for the update (3). Starting from the definition

$$\lambda_{t+1} = \arg\max_{\lambda \in \mathcal{X}} \{\eta \langle g_t, \lambda \rangle + B_{\Psi_t}(\lambda, \lambda_t)\}$$

where we remark the definition of Bregman divergence

$$B_{\Psi_t}(\lambda, \lambda_t) = \Psi_t(\lambda) - \Psi_t(\lambda_t) - \langle \nabla \Psi_t(\lambda_t), \lambda - \lambda_t \rangle$$

Also in this case, to find the $\arg\max$ we derive the update formula and set it to be zero

$$\begin{aligned} \frac{\partial}{\partial \lambda} \eta \langle g_t, \lambda \rangle + \frac{\partial}{\partial \lambda} B_{\Psi_t}(\lambda, \lambda_t) = 0 \\ \eta\, g_t + \frac{\partial}{\partial \lambda} [\Psi_t(\lambda) - \Psi_t(\lambda_t) - \langle \nabla \Psi_t(\lambda_t), \lambda - \lambda_t \rangle] = 0 \\ \eta\, g_t + H_t\, \lambda - \frac{\partial}{\partial \lambda} \langle \nabla \Psi_t(\lambda_t), \lambda - \lambda_t \rangle = 0 \end{aligned}$$

Considering the first term of the scalar product, we have to evaluate the first order taylor model of the proximal function Ψ_t at the known quantity λ_t. It is easy to see that by using the derivation in the first paragraph and the fact that H_t is diagonal

$$\nabla\Psi_t(\lambda_t) = \nabla\frac{1}{2}\langle\lambda_t, H_t\lambda_t\rangle = H_t\lambda_t$$

Putting this into equation

$$\frac{\partial}{\partial\lambda}\langle H_t\lambda_t, \lambda - \lambda_t\rangle = \frac{\partial}{\partial\lambda}\sum_{i=1}^{n} h_{ii}\lambda_{t,i}(\lambda_i - \lambda_{t,i})$$

And so is easy to see that

$$\frac{\partial}{\partial\lambda}\langle H_t\lambda_t, \lambda - \lambda_t\rangle = \begin{bmatrix} \frac{\partial}{\partial\lambda_1}\langle H_t\lambda_t, \lambda - \lambda_t\rangle \\ \frac{\partial}{\partial\lambda_2}\langle H_t\lambda_t, \lambda - \lambda_t\rangle \\ \cdots \\ \frac{\partial}{\partial\lambda_n}\langle H_t\lambda_t, \lambda - \lambda_t\rangle \end{bmatrix} = \begin{bmatrix} h_{11}\lambda_{t,1} \\ h_{22}\lambda_{t,2} \\ \cdots \\ h_{nn}\lambda_{t,n} \end{bmatrix} = H_t\lambda_t$$

And finally putting this into the main problem, we get:

$$\eta\, g_t + H_t\,\lambda - H_t\lambda_t = 0 \quad \Longrightarrow\; H_t\,\lambda = H_t\lambda_t - \eta\, g_t$$
$$\lambda = H_t^{-1}\left[H_t\lambda_t - \eta\, g_t\right] = \lambda_t - \eta\, H_t^{-1}\, g_t$$

B Appendix B: subgradient computation

Utterly following the definition, we state that s is a subgradient of ψ at x

$$\psi(\lambda) \leq \psi(x) + s^{\intercal}(\lambda - x) \quad \forall\lambda \in \mathbb{R}^n$$

which reordering the term can be written as

$$s^{\intercal}(\lambda - x) \geq \psi(\lambda) - \psi(x) \quad \forall\lambda \in \mathbb{R}^n$$

Either we can solve the above complex inequality with unknown variables s and λ, or better we can apply the following reasoning. The subdifferential is the set containing all the numbers in the interval $[a, b]$ such that

$$a = \lim_{\lambda\to\lambda_0^-} \frac{\psi(\lambda) - \psi(\lambda_{t-1})}{\lambda - \lambda_{t-1}}$$
$$b = \lim_{\lambda\to\lambda_0^+} \frac{\psi(\lambda) - \psi(\lambda_{t-1})}{\lambda - \lambda_{t-1}}$$

where the $^+$ and $^-$ portion must have a certain value greater than zero (order of 10^{-10}).

C Appendix C: γ computation

Deflection is a way to change the direction of the computed subgradient by using a convex combination

$$d^i = \gamma^i g^i + (1 - \gamma^i)d^{i-1}$$

where g^i is the subgradient at iteration i and d^{i-1} is the direction computed at the previous step.
To find the best γ^i at iteration i, we could solve the problem

$$\gamma^i \in \arg\min\left\{\|\gamma g^i + (1 - \gamma)d^{i-1}\|^2 : \gamma \in [0, 1]\right\}$$

Hence we have to solve (we omit the iteration for the sake of visualization)

$$\frac{\partial}{\partial \gamma}\|\gamma g + (1-\gamma)d\|^2$$

which can be rewritten as

$$\frac{\partial}{\partial \gamma}\sum_{k=1}^{n}(\gamma_k g_k + (1-\gamma_k)d_k)^2 = 0$$

which result in the following vector

$$\begin{bmatrix} \frac{\partial}{\partial \gamma_1}\sum_{k=1}^{n}(\gamma_k g_k + (1-\gamma_k)d_k)^2 \\ \frac{\partial}{\partial \gamma_2}\sum_{k=1}^{n}(\gamma_k g_k + (1-\gamma_k)d_k)^2 \\ \vdots \\ \frac{\partial}{\partial \gamma_n}\sum_{k=1}^{n}(\gamma_k g_k + (1-\gamma_k)d_k)^2 \end{bmatrix} = \mathbf{0} \implies \begin{bmatrix} \frac{\partial}{\partial \gamma_1}(\gamma_1 g_1 + (1-\gamma_1)d_1)^2 \\ \frac{\partial}{\partial \gamma_2}(\gamma_2 g_2 + (1-\gamma_2)d_2)^2 \\ \vdots \\ \frac{\partial}{\partial \gamma_n}(\gamma_n g_n + (1-\gamma_n)d_n)^2 \end{bmatrix} = \mathbf{0} \implies$$

$$\begin{bmatrix} 2(\gamma_1 g_1 + (1-\gamma_1)d_1)(g_1 - d_1) \\ 2(\gamma_2 g_2 + (1-\gamma_2)d_2)(g_2 - d_2) \\ \vdots \\ 2(\gamma_n g_n + (1-\gamma_n)d_n)(g_n - d_n) \end{bmatrix} = \mathbf{0}$$

Isolating each γ term, putting back the iteration i. we obtain the following compact value for γ at iteration i:

$$\gamma^i = \frac{d^{i-1}(d^{i-1} - g^i)}{(g^i - d^{i-1})^2}$$