Homework 1

Convex Optimization

Matthieu Boyer

20 février 2025

1 Exercises

Convex Sets

- Let $\mathcal{C} = \left\{x \in \mathbb{R}_+^2 \mid x_1 x_2 \geq 1\right\}$. We can see that $\mathcal{C} = \left\{(x_1, x_2) \in \mathbb{R}_{+*}^2 \mid x_1 \geq \frac{1}{x_2}\right\}$. Thus, $\mathcal{C}$ is the epigraph of $x \mapsto \frac{1}{x}$ which is convex since its second derivative is $x \mapsto \frac{2x}{x^4}$. Thus, $\boxed{\mathcal{C} \text{ is convex.}}$
- In general, $\boxed{\text{this set is not convex.}}$ Indeed, for sets in $\mathbb{R}$, take $S = \{-1, 1\}, T = \{0\}$. Then the set of points closer to S than to T is $\{x \in \mathbb{R} \mid x \leq -0.5 \vee x \geq 0.5\}$ which is not an interval and thus not convex.
- Let us take $x_1, x_2 \in \{x \mid x + S_2 \subseteq S_1\}$. For $x = \lambda x_1 + (1 - \lambda)\, x_2$ consider $x + y$ for $y \in S_2$:

$$x + y = \lambda x_1 + (1 - \lambda)\, x_2 + y = \lambda(x_1 + y) + (1 - \lambda x_2 + y)$$

  Since $x_i + S_2 \subseteq S_1$ for $i = 1, 2$, and since S_1 is convex, for all $y \in S_2$ the above sum is in S_1 and thus $x + S_2 \subseteq S_2$. Finally, $\boxed{\text{our set is convex.}}$
- Let $x_1, x_2 \in \{x \mid \exists y \in S_2, x + y \in S_1\}$. Then let y_1, y_2 be associated points to x_1, x_2. For $x = \lambda x_1 + (1 - \lambda)\, x_2$ and $y = \lambda y_1 + (1 - \lambda) y_2$. Then, since S_2 is convex, $y \in S_2$. Moreover :

$$x + y = \underbrace{\lambda \underbrace{(x_1 + y_1)}_{\in S_1} + (1 - \lambda) \underbrace{(x_2 + y_2)}_{\in S_1}}_{\in S_1 \text{ since } S_1 \text{ is convex}}$$

  Then, $\boxed{\text{our set is convex.}}$

Convex Functions

- The hessian of f at (x, y) is the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. On $\mathbb{R}^2$ this matrix is not positive nor negative and thus the function is neither convex, or concave. However, the sub-level sets for α are convex if and only if $\alpha \geq 0$ (from our first example) and thus $\boxed{\text{the function is neither quasi-concave nor quasi-convex.}}$
- On $\mathbb{R}_{+*}^2$, the hessian of the function is positive semidefinite and thus $\boxed{\text{the function is convex.}}$
- The hessian matrix of f is the matrix $\begin{pmatrix} 0 & -\frac{1}{x_2^2} \\ -\frac{1}{x_2^2} & \frac{2x_1}{x_2^3} \end{pmatrix}$ Since its determinant is < 0 the Hessian is not positive semi-definite and the function is not convex. Moreover, since its determinant is < 0 and the hessian is symmetric real, it must have a strictly positive eigenvalue, and thus it is not concave. However, its sublevel sets are defined by the equations $x_1 \leq \alpha x_2$ and are thus convex (since half-planes). Similarly the sub-level sets are concave. Thus, $\boxed{f \text{ is quasi-convex and quasi-concave}}$.

- Let us define the Löwner order $\preceq$ as the partial order defined by the convex cone of positive semi-definite matrices. We know that :

$$A \preceq B \Rightarrow B^{-1} \preceq A^{-1}, \forall A, B \in S_{++}^n$$

From this, we know that for all $t \leq 1$:

$$\left((1-t)\,X + tY\right)^{-1} \preceq (1-t)\,X^{-1} + tY^{-1}$$

By linearity of the trace, we can now see that $\boxed{X \mapsto \mathrm{Tr}(X^{-1}) \text{ is convex}}$.

Fenchel Conjugate

- We have :

$$\boxed{f^*(y) = \sup_x \left({}^{\mathsf{t}}xy - \|x\|\right) = \begin{cases} 0 & \text{if } \|y\|_* \leq 1 \\ \infty & \text{otherwise} \end{cases}}$$

To show this, let y such that $\|y\|_* = \sup_{\|x\| \leq 1} \|{}^{\mathsf{t}}xy\| \leq 1$. Then by Cauchy-Schwarz inequality, we know that ${}^{\mathsf{t}}xy \leq \|x\| \, \|y\|_* \leq \|x\|$ and the term in the supremum is always ≤ 0. If $\|y\|_* > 1$ however, then there exists z such that $\|z\| \leq 1$ such that ${}^{\mathsf{t}}zy > 1$. Taking $x = tz$ in the supremum, we know that $f^*(y) \geq t\left({}^{\mathsf{t}}zy - \|z\|\right)$ which goes to infinity with $t \to \infty$. Thus, the fenchel conjugate of a norm is the convex indicator of the unit ball of the dual norm.

For the case of the square of the ℓ^2 norm, we want to compute :

$$f^*(y) = \sup_x {}^{\mathsf{t}}yx - {}^{\mathsf{t}}xx = {}^{\mathsf{t}}\left(s - x\right)x$$

Fenchel-Young's theorem gives us :

$$f^*\left(y\right) = {}^{\mathsf{t}}yx - f(x) \Leftrightarrow y \in \partial f(x) \Leftrightarrow y = 2x$$

Thus :

$$\boxed{f^*\left(y\right) = \frac{1}{4}{}^{\mathsf{t}}yy}$$

- Let us denote the infimal convolution of g, h by $g \square h$. Then we will show that :

$$\boxed{\left(g \square h\right)^* = \left(g^* + h^*\right)}$$

Note that :

$$\begin{aligned}
\left(g \square h\right)^*\left(\alpha\right) =& \sup_{x,y} \left\{{}^{\mathsf{t}}x\alpha - g(y) - h(x-y)\right\} \\
=& \sup_{x_1, x_2} \left\{{}^{\mathsf{t}}\left(x_1 + x_2\right)\alpha - g(x_1) - h(x_2)\right\} \\
=& \sup_{x_1, x_2} \left\{{}^{\mathsf{t}}x_1\alpha - g(x_1)\right\} + \sup_{x_2} \left\{{}^{\mathsf{t}}x_2\alpha - h(x_2)\right\} \\
=& g^* + h^*
\end{aligned}$$

Given $g = \|\cdot\|_1$ and $h = \frac{1}{2\alpha}\|\cdot\|_2^2$. Let $x = u + v$. Then :

$$f(x) = \inf_v \left\{g(x-v) + h(v)\right\}$$

Substituting the expressions, we get :

$$\begin{aligned} f(x) &= \inf_v \left\{ \|x-v\|_1 + \frac{1}{2\alpha}\|v\|_2^2 \right\} \\ &= \sum_{i=1}^n \inf_{v_i} \left\{ |x_i - v_i| + \frac{1}{2\alpha} v_i^2 \right\} \end{aligned}$$

We will now compute the infima independently. We have two cases :

1. $v_i \leq x_i$: Let $\varphi(v_i) = x_i - v_i + \frac{1}{2\alpha}v_i^2$. We want to find a minimum for φ, which is found at $v_i = \alpha$. This solution is valid if $\alpha \leq x_i$.
2. $v_i > x_i$: Let $\varphi(v_i = \frac{1}{2\alpha} + v_i - x_i$. We want to find a minimum for φ, which is at $v_i = -\alpha$. This solution is valid if $\alpha > x_i$.

Then, we have three possible cases for the optimal v_i :

1. If $x_i \geq \alpha$, $v_i = \alpha$
2. If $x_i \leq -\alpha$, $v_i = -\alpha$
3. If $-\alpha < x_i < \alpha$, $v_i = x_i$.

Plugging this into f :

$$\boxed{f(x) = \sum_{i=1}^n \left(\begin{cases} \frac{1}{2\alpha}x_i^2 & |x_i| \leq \alpha \\ |x_i - \alpha| - \frac{\alpha}{2} & |x_i| > \alpha \end{cases} \right)}$$

Now, clearly, $f = g \Box h$ is a convex function (as the infimum of two convex functions (if $\alpha > 0$)). Moreover, f is clearly lower semi-continuous from its expression. Now, we have $\boxed{f^{**} = f}$ from the Fenchel-Moreau theorem (proved below) :

Théorème 1.1 The biconjugate of f is the largest lower semi-continuous convex function below than f.

Démonstration. Let $x \in \mathbb{R}^n$. For all y, the directional derivative :

$$\partial_y f(x) = \lim_{\mathrm{d}t \to 0} \frac{f(x + \mathrm{d}ty) - f(x)}{\mathrm{d}t}$$

is a sublinear as a function of y. From the Hahn-Banach theorem, there exists $\tilde{\partial f} \in (\mathbb{R}^n)^* = \mathbb{R}^n$ such that :

$$\langle \tilde{\partial f}, \cdot \rangle \leq \partial_y f(x) \leq f(x+y) - f(x), \forall y \in \mathbb{R}^n$$

Then :

$$f(x) + f^*\left(\tilde{\partial f}\right) = \langle \tilde{\partial f}, x \rangle$$

which completes our proof that $f^{**} = f$. ∎

- We want to compute :

$$\ell^*(y) = \sup_{z \in \mathbb{R}} \{yz - \ell(z)\} \text{ where } \ell : z \mapsto \log(1 + e^z)$$

We differentiate $\varphi(z)$ the function in the supremum :

$$\varphi'(z) = y - \frac{1}{1 + e^{-z}}$$

Then, we find $z = \log\left(\frac{y}{1-y}\right)$, which only makes sense for $y \in]0,1[$. Finally, we compute :

$$\begin{aligned}\ell^*(y) =& y\log\left(\frac{y}{1-y}\right) + \log(1-y)\\ =& -y\log(1-y) + y\log(y) + \log(1-y)\\ =& (1-y)\log(1-y) + (y)\log(y)\end{aligned}$$

In the end,

$$\boxed{(1-y)\log(1-y) + (y)\log(y)}$$

Duality

- The Lagrangian of the problem is :

$$\mathcal{L}(x,\lambda) = \frac{1}{2}\|Ax-b\|_2^2 + \alpha\|x\|_1 - {}^{\mathsf{t}}\lambda\,(Ax-b) = \frac{1}{2}\|Ax-b\|_2^2 + \alpha\|x\|_1 - {}^{\mathsf{t}}\lambda Ax + {}^{\mathsf{t}}\lambda b$$

Then, by separating terms involving x :

$$\mathcal{L}(x,\lambda) = \frac{1}{2}\|b\|_2^2 - \frac{1}{2}\|\lambda - b\|_2^2 + {}^{\mathsf{t}}\lambda b + \alpha\|x\|_1 - \left({}^{\mathsf{t}}\lambda A\right)x$$

In solving $\max_\lambda \min_x \mathcal{L}(x,\lambda)$, the inner minimization on x only depends on $\|x\|_1$ and $({}^{\mathsf{t}}\lambda A)\,x$. Minimizing it results in $\|{}^{\mathsf{t}}A\lambda\|_\infty \le \alpha$. Finally, we get that :

$$\boxed{\max_{\lambda\in\mathbb{R}^m} -\frac{1}{2}\|\lambda-b\|_2^2 + \frac{1}{2}\|b\|_2^2 - \inf_{\{\|\cdot\|_\infty \le 1\}}\left(\frac{{}^{\mathsf{t}}A\lambda}{\alpha}\right) \text{ is a dual problem for the LASSO}}$$

- We start with :

$$\min_{w,w_1,\dots,w_m\in\mathbb{R}^n}\left\{\frac{1}{n}\sum_{i=1}^m h_i(w_i) + \frac{\lambda}{2}\|w\|_2^2 \;\middle|\; w_i = w, \forall i \in [\![1,m]\!]\right\}$$

Its Lagrangian is :

$$\mathcal{L}(w,w_1,\dots,w_m,v_1,\dots,v_m) = \frac{1}{n}\sum_{i=1}^m h_i(w_i) + \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^m {}^{\mathsf{t}}v_i(w-w_i) = \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^m {}^{\mathsf{t}}v_i w + \sum_{i=1}^m \frac{1}{n}h_i(w_i) - {}^{\mathsf{t}}v_i w_i$$

For the minimization step, we minimize for each of the w_i and for w. Let $g_i(v_i)$ be the minimized value of $\frac{1}{n}h_i(w_i) - {}^{\mathsf{t}}v_i w_i$. For w, we want to minimize $\frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^m {}^{\mathsf{t}}v_i w$, which is quadratic in w. Computing its minimum we find $\frac{-1}{2\lambda}\|\sum_{i=1}^m v_i\|_2^2$. Finally, the dual problem is :

$$\boxed{\max_{v_1,\dots,v_m}\sum_{i=1}^m g_i(v_i) - \frac{1}{2\lambda}\left\|\sum_{i=1}^m v_i\right\|_2^2}$$

For logistic regression, we have :

$$h_i : w \mapsto \log\left(1+\exp\left(-y_i{}^{\mathsf{t}}x_i w\right)\right)$$

Then we have :

$$g_i(v_i) = \min_{w_i}\left\{\frac{1}{n}\log\left(1+\exp\left(-y_i{}^{\mathsf{t}}x_i w_i\right)\right) - {}^{\mathsf{t}}v_i w_i\right\}$$

Since h_i is differentiable, we can compute its gradient :

$$\nabla h_i\left(w_i\right) = \frac{1}{n} \cdot \frac{-y_i x_i}{1 + \exp\left({y_i}^{\mathsf{t}} x_i w_i\right)}$$

which gives us a minimality condition :

$$\frac{-y_i x_i}{1 + \exp\left({y_i}^{\mathsf{t}} x_i w_i\right)} = n v_i$$

which rearranges to :

$$w_i = \frac{-y_i \log\left(\frac{-{y_i}^{\mathsf{t}} v_i x_i - n\|v_i\|_2^2}{n\|v_i\|_2^2}\right)}{\|x_i\|_2^2} x_i = \frac{-y_i t}{\|x_i\|_2^2} x_i$$

Inputing this into ${y_i}^{\mathsf{t}} x_i w_i$:

$$ {y_i}^{\mathsf{t}} x_i w_i = + \underbrace{y_i^2}_{=1} \frac{\overbrace{{}^{\mathsf{t}} x_i x_i}^{=1}}{\|x_i\|_2^2} t = t$$

Thus :

$$g_i(w_i) = \frac{1}{n} \log\left(-\frac{{y_i}^{\mathsf{t}} v_i x_i}{\|v_i\|_2^2}\right) - \frac{{y_i}^{\mathsf{t}} v_i w_i}{\|x_i\|_2^2} \log\left(-1 - \frac{{y_i}^{\mathsf{t}} v_i x_i}{\|v_i\|_2^2}\right)$$

Finally our dual problem is :

$$\boxed{\max_{v_1,\dots,v_m} \sum_{i=1}^{m} \frac{1}{n} \log\left(-\frac{{y_i}^{\mathsf{t}} v_i x_i}{\|v_i\|_2^2}\right) - \frac{{y_i}^{\mathsf{t}} v_i w_i}{\|x_i\|_2^2} \log\left(-1 - \frac{{y_i}^{\mathsf{t}} v_i x_i}{\|v_i\|_2^2}\right) - \frac{1}{2\lambda} \left\|\sum_{i=1}^{m} v_i\right\|_2^2}$$

and I will not be trying to find a closer form manually.

- We consider the problem :

$$\min_{X \in \mathbb{S}^n} \left\{\operatorname{Tr}\left(A_0 X\right) \mid X \succeq 0, \operatorname{Tr}\left(A_1 X\right) = b_1, \dots, \operatorname{Tr}\left(A_m X\right) = b_m\right\}$$

Its Lagrangian is (for $\lambda_i > 0, S \succeq 0$) :

$$\mathcal{L}\left(X, \lambda_1, \dots, \lambda_m, S\right) = \operatorname{Tr}\left(A_0 X\right) + \sum_{i=1}^{m} \lambda_i \left(\operatorname{Tr}\left(A_i X\right) - b_i\right) + \operatorname{Tr}\left({}^{\mathsf{t}} S X\right) = \operatorname{Tr}\left(\left(A_0 + \sum_{i=1}^{m} \lambda_i A_i + S\right) X\right) - \sum_{i=1}^{m} \lambda_i b_i$$

Minimizing everything in X can be done since everything is differentiable in X :

$$\nabla \mathcal{L} = {}^{\mathsf{t}} A_0 + \sum_{i=1}^{m} \lambda_i {}^{\mathsf{t}} A_i + 2SX = A_0 + \sum_{i=1}^{m} \lambda_i A_i + S$$

which is 0 when :

$$S = -\left(A_0 + \sum_{i=1}^{m} \lambda_i A_i\right)$$

Finally we get the dual problem :

$$\boxed{\max_{\lambda_1,\dots,\lambda_m,S} \left\{-\sum_{i=1}^{m} \lambda_i b_i \;\middle|\; S \succeq 0, S = -A_0 - \sum_{i=1}^{m} \lambda_i A_i\right\}}$$

Strong duality holds only if the Karush-Kuhn-Tucker conditions are satisfied. Since $\operatorname{Tr}\left(S_* X_*\right)$ is a rewriting of complementary slackness, clearly strong duality implies $\operatorname{Tr}(S_* X_*) = 0$. If we have $\operatorname{Tr}(S_* X_*) = 0$ for the optimal solutions, in particular, both the primal and dual problems are attained and the relaxation term vanishes. Finally :

$$\boxed{\text{Strong Duality holds if and only if } \operatorname{Tr}(X_* S_*) = 0}$$

2 Problem

We consider the following problem :

$$\min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|_2^2 + \alpha \sum_{i=1}^m \max\left\{0, 1 - {}^{\mathsf{t}} w x_i y_i\right\} \tag{1}$$

2.1 Dual problem.

Considering the constraints, we have :

$$s_i \geq 1 - y_i {}^{\mathsf{t}} w x_i \wedge s_i \geq 0$$

and thus, minimizing in s, for w fixed gives :

$$s_i = \max\left\{0, 1 - y_i {}^{\mathsf{t}} w x_i\right\}$$

which proves the equivalent reformulation :

$$\begin{aligned} \min_{w \in \mathbb{R}^n, s \in \mathbb{R}^m} \frac{1}{2} \|w\|_2^2 + \alpha \sum_{i=1}^m s_i \\ \text{s. t. } {}^{\mathsf{t}} w x_i y_i \geq 1 - s_i \\ s_i \geq 0 \end{aligned}$$

Computing the dual problem needs two constraints :

$$\lambda_i \geq 0 \text{ for } {}^{\mathsf{t}} w X_i \geq 1 - s_i \text{ and } \mu_i \geq 0 \text{ for } s_i \geq 0$$

The Lagrangian is thus :

$$\mathcal{L}(w, s, \lambda, \mu) = \frac{1}{2} \|w\|_2^2 + \alpha \sum_{i=1}^m s_i - \sum_{i=1}^m \lambda_i \left({}^{\mathsf{t}} w X_i - 1 + s_i\right) - \sum_{i=1}^m \mu_i s_i$$

Then :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^m \lambda_i X_i = 0 \Rightarrow w = X\lambda \\ \frac{\partial \mathcal{L}}{\partial s_i} = \alpha - \lambda_i - \mu_i = 0 \Rightarrow \lambda_i \leq \alpha \end{aligned}$$

Finally, subtituting, the dual problem is :

$$\max_{0 \leq \lambda \leq \alpha} \mathcal{D}(\lambda) = -\frac{1}{2} {}^{\mathsf{t}} \lambda {}^{\mathsf{t}} X X \lambda + \sum_{i=1}^m \lambda_i$$

2.2 Algorithms

Everything in this section has been implemented using Python 3.12. Preamble :

```
import functools

from random import randint
import numpy as np
import numpy.linalg as npl


def projection(lower, upper, value):
    return min(upper, max(value, lower))


def d(lam, X):
    return -(1/2) * lam.T.dot(X.T).dot(X).dot(lam) + sum(lam)


def gradientD(lam, X):
    return np.array([1 for _ in lam]) - X.T.dot(X).dot(lam)
```

- Implementation for Gradient Ascent

```
def gradient_ascent_dual_svm(dataset, alpha, iterations):
    """
    :param alpha: parameter of the SVM problem
    :param dataset: List of couples x_{i}:np.array, y_{i}:+-1 representing our classes
    :param iterations: Number of iterations to compute
    :return:
    """
    proj = functools.partial(projection, lower=0.0, upper=alpha)
    X = np.zeros((n:=len(dataset[0][0]), m:=len(dataset)))
    for i in range(m):
        X[i] = dataset[i][0] * dataset[i][1]

    lam = 0
    h = max(npl.eigvalsh(X))
    for _ in range(iterations):
        olam = lam
        lam = map(proj, olam + h * gradientD(olam, X))
        while d(lam, X) > d(olam, X) + (1/(h * 2)) * (npl.norm(lam - olam) ** 2):
            h = h /2
            lam = map(proj, olam + h * gradientD(olam, X))

    return X.dot(lam)
```

- Implemenation for Randomized Coordinate Ascent

```
def randomized_coordinate_ascent(dataset, alpha, iterations):
    proj = functools.partial(projection, lower=0.0, upper=alpha)
    lam = np.zeros(m:= len(dataset))
    n = len(dataset[0][0])
    w = np.zeros(n)
    for _ in range(iterations):
        i = randint(0, m - 1)
        olam = lam[i]
        lam = lam.copy()
        lam[i] = proj(olam + (1 - (yi := dataset[i][1]) * (xi := dataset[i][0]).T.dot(w))/(npl.norm(xi)**2))
        w = w + yi * xi.dot(lam[i] - olam)
    return w
```

Let us prove the algorithm leads to optimization, by optimizing on one dimension at a time. At each iteration, only one dimension λ_{i_k} is modified, and the rest remain the same. Since the part of $D(\lambda)$ dependent on λ_{i_k} is

$$-\frac{1}{2}\left\|x_{i_k}\right\|^2 \lambda_{i_k}^2 + \left(1 - {y_{i_k}}^{\mathrm{t}} x_{i_k} w\right) \lambda_{i_k}$$

then taking

$$\lambda_{i_k}^{(k+1)} = \lambda_{i_k}^{(k)} + \frac{(1 - y_{i_k}{}^{\mathsf{t}}x_{i_k} w)}{\|x_{i_k}\|_2^2}$$

maximizes the above quadratic term and the projection ensures the solution is feasible, leading to an exact maximization along λ_{i_k} by the properties of quadratic forms.

I have not done experiments using datasets for compatibility reasons. However, I originally intended to include a full working gradient descent in the code of this report, without external programs. This project was aborted due to time consumption issues, here is the main code. The only non-working part should be the computation of a maximum eigenvalue of ${}^{\mathsf{t}}XX$.

```
\usepackage{fp}
\usepackage{pgffor}
\usepackage{xstring}

\newcommand{\ProjectScalar}[2]{%
    \FPiflt{#1}{0}%
        \FPset\projection{0}%
    \else%
        \FPifgt{#1}{#2}%
            \FPset\projection{#2}%
        \else%
            \FPset\projection{#1}%
        \fi%
    \fi%
}

\newcommand{\MatrixVectorMultiply}[2]{%
    \renewcommand{\result}{}
    \foreach \row in #1 {%
        \FPset\rowSum{0}
        \foreach \xi \lambdaElem in \row #2 {%
            \FPmul\product{\xi}{\lambdaElem}
            \FPadd\rowSum{\rowSum}{\product}
        }
        \xdef\result{\result\rowSum\space}
    }
}
```

Please turn over for the rest of the code.

```
\newcommand{\DualProjectedGradientAscent}[4]{%
    % #1: Data matrix X (comma-separated rows with elements separated by spaces)
    % #2: y vector (space-separated)
    % #3: alpha (regularization parameter)
    % #4: max_iterations (number of iterations)
    \def\matrixX{#1}
    \def\yVector{#2}
    \FPset\alpha{#3}
    \FPset\maxIter{#4}
    \FPset\maxEigenvalue{10} % Compute max eigenvalue of X^T X (placeholder)
    \FPmul\unh{\alpha}{\maxEigenvalue}
    \FPdiv\hInitial{1}{\unh}
    \def\rowCount{0}
    \def\colCount{0}
    \StrCount{\matrixX}{,}[\rowCount]
    \FPadd\rowCount{\rowCount}{1}
    \StrCount{\matrixX}{ }[\colCount]
    \FPdiv\colCount{\colCount}{\rowCount}
    \newcommand{\lambdaVector}{}
    \foreach \col in {1,...,\colCount}{\xdef\lambdaVector{\lambdaVector0\space}}
    \newcommand{\gradientVector}{}
    \newcommand{\objective}{0}
    \FPset\h{\hInitial}
    \FPset\iter{0}
    \loop
        \ifnum\iter<\maxIter
            \MatrixVectorMultiply{\matrixX}{\lambdaVector}
            \renewcommand{\gradientVector}{}
            \foreach \row in \matrixX {%
                \FPset\rowSum{0}
                \foreach \xi in \row {%
                    \FPmul\product{\xi}{\row}
                    \FPadd\rowSum{\rowSum}{\product}
                }
                \xdef\gradientVector{\gradientVector\rowSum\space}
            }
            \renewcommand{\updatedGradient}{}
            \foreach \gradElem \yElem in \gradientVector \yVector {%
                \FPadd\sumValue{\gradElem}{\yElem}
                \xdef\updatedGradient{\updatedGradient\sumValue\space}
            }
            \renewcommand{\prevLambda}{\lambdaVector}
            \FPset\prevObjective{\objective}
            \renewcommand{\lambdaVector}{}
            \foreach \lambdaElem \gradElem in \prevLambda \updatedGradient {%
                \FPmul\stepUpdate{\h}{\gradElem}
                \FPadd\newValue{\lambdaElem}{\stepUpdate}
                \ProjectScalar{\newValue}{\alpha}
                \xdef\lambdaVector{\lambdaVector\projection\space}
            }
            \MatrixVectorMultiply{\matrixX}{\lambdaVector}
            \renewcommand{\objective}{0}
            \foreach \lambdaElem \resultElem in \lambdaVector \result {%
                \FPmul\product{\lambdaElem}{\resultElem}
                \FPadd\objective{\objective}{\product}
            }
            \FPmul\objective{-0.5}{\objective}
            \foreach \lambdaElem \yElem in \lambdaVector \yVector {%
                \FPmul\product{\lambdaElem}{\yElem}
                \FPadd\objective{\objective}{\product}
            }

            \FPiflt{\objective}{\prevObjective}
                \FPdiv\h{\h}{2}
                \renewcommand{\lambdaVector}{\prevLambda}
            \else
                \FPadd\iter{\iter}{1}
            \fi
    \repeat
    \MatrixVectorMultiply{\matrixX}{\lambdaVector}
    \textbf{Final weights:} \result
}
% Matrix X: "1 2, 3 4" (2x2 matrix), y vector = "1 1", alpha = 1, 10 iterations
\DualProjectedGradientAscent{1 2, 3 4}{1 1}{1}{10}
```