Optimizari
Laborator 4: Recuperarea imaginilor. Gradient proiectat. Gradient Conditional. CVX.

In acest laborator rezolvam **problema cu constrangeri** provenita din problema completarii matricei (engl. *Matrix completion*) pentru a recupera o imagine ce are pixeli lipsa. Vom rezolva problem folosind CVX, gradientul proiectat si gradientul conditional.

1 Introducere in toolbox-ul CVX

CVX-ul este un sistem de modelare pentru construirea și rezolvarea problemelor convexe. Acesta este implementat in Matlab, transformand eficient Matlab-ul intr-un limbaj de modelare optimizat. CVX-ul accepta pe langa Matlab, doua *solutii gratuite*, **SeDuMi** [3] si **SDPT3** [4], acestea fiind incluse cu distribuția CVX. Alaturi de optiunile gratuite, CVX-ul accepta si doua *solvere comerciale*, **Gurobi** [5] si **MOSEK** [6].

1.1 Instalare

1. Se descarca pachetul standard CVX potrivit sistemului de operare utilizat de la urmatoarea adresa `http://cvxr.com/cvx/download/` [1]
2. Se dezarhiveaza intr-un director dorit (diferit de directorul toolbox al Matlab-ului)
3. Se porneste Matlab-ul;
4. Se comuta directorul curent in directorul unde am realizat dezarhivarea si se executa in consola comanda **cvx_setup**.

1.2 Elemente de baza

- Orice program CVX se scrie in interiorul unei functii Matlab.
- Programele CVX se delimiteaza de codul Matlab cu comenzile **cvx_begin** si **cvx_end**.
- Valorile variabilelor create in portiunea de cod Matlab se pot folosi ca parametri in problemele de optimizare rezolvate cu CVX.

[1] La aceasta adresa gasiti si un ghid de instalare mai detaliat

- Variabilele CVX se declara folosind comanda **variable** si specificarea dimensiunii variabilei
  - Exemple:
    Vectorul $x \in \mathbb{R}^n$: `variable x(n)`
    Matricea $A \in \mathbb{R}^{n \times m}$: `variable A(n,m)`
  - o varietate de optiuni aditionale pentru precizarea structurii matriceale sunt disponibile la adresa `http://cvxr.com/cvx/doc/basics.html`
    Exemplu: O matrice simetrica se declara
    `variable A(10,10) symmetric`
- La inceputul oricarui program CVX se definesc variabilele de decizie si dimensiunile acestora.
- Declararea *functiei obiectiv* a problemei de optimizare necesita precizarea tipului de problema (e.g. minimizare, maximizare) prin intermediul cuvintelor cheie **minimize** si **maximize**
  - Este necesar ca functia obiectiv sa fie <u>convexa</u> cand folosim <u>minimize</u> si <u>concava</u> cand folosim <u>maximize</u>. In caz contrar, pachetul CVX va furniza un mesaj de eroare corespunzator.
- *Constrangerile* suportate de modelele CVX sunt cele de *egalitate* (liniare) impuse prin operatorul == si de *inegalitate* impuse de operatorii <= si >=.
  - Pentru *constrangeri de tip box* (lant de inegalitati) este disponibila sintaxa $l <= x <= u$.

Pentru asimilarea facila a acestor reguli sa ne uitam la exemplele de implementare in CVX din urmatoarea sectiune.

1.3 Exemple

Metoda celor mai mici patrate constransa
Fie problema de optimizare cu constrangeri:

$$\min_{x \geq 1} \|Ax - b\|.$$

Luand o matrice A de dimensiune $m \times n$ (ex: $A = \text{randn}(m, n)$) si un vector b in Matlab, avem urmatorul cod CVX pentru a gasi solutia problemei:

```
m =5; n=10;
A = randn(m,n);
b = randn(m,1);
cvx_begin
variable x(n)
minimize(norm(A*x-b))
subject to
x >= 1
cvx_end
```

Observam in fereastra de comanda urmatoarele informatii odata ce am rulat codul de mai sus:

```
Calling SDPT3 4.0: 16 variables, 5 equality constraints
------------------------------------------------------------

 num. of constraints =  5
 dim. of socp   var  =  6,   num. of socp blk  =  1
 dim. of linear var  = 10
```

Remarcam ca suntem informatii ce solver a fost chemat. In cazul de mai sus a fost SDPT3. Acest lucru se poate schimba cu comanda **cvx_solver** urmat de numele solverului. Putem observa de asemenea ca numarul de constrangeri este egal cu m si anume 5, iar dimensiunea variabilei liniare cu n, respectiv 10.

```
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|7.9e+00|1.6e+01|1.0e+03| 6.343522e+00  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.236|4.0e-06|1.2e+01|9.2e+02| 7.245086e+01  9.552675e+00| 0:0:00| chol  1  1
```

Dupa aceasta sectiune sunt afisate informatii detaliate ale iteratiilor, cum ar fi: valoarea pasului la problema primal si duala, valoarea functiei obiectiv primala si duala, timpul CPU, etc De asemenea este afisat criteriul de oprire, si tot odata precizia cu care a fost calculata solutia. Acest parametru se poate schimba din setari sau adaugand comanda **cvc_precision high/medium/best** sau **low**. In functie de pozitionarea aceasteia, ea are efect global (daca este inainte de cvx_begin) sau local (intre cvx_begin si cvx_end)

```
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 12
 primal objective value =  8.45423413e-09
 dual   objective value =  1.09498700e-09
 gap := trace(XZ)       = 1.42e-08
 relative gap           = 1.42e-08
 actual relative gap    = 7.36e-09
 rel. primal infeas (scaled problem)   = 5.24e-14
 rel. dual     "        "       "      = 4.89e-11
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.3e+02, 2.8e-10, 1.0e+00
 norm(A), norm(b), norm(C) = 7.7e+00, 7.3e+00, 2.0e+00
 Total CPU time (secs)  = 0.59
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 7.4e-14  0.0e+00  4.9e-11  0.0e+00  7.4e-09  1.4e-08
-------------------------------------------------------------------
```

Detalierea iteratiilor, este urmata de un rezumat ce cuprinde: numarul total de iteratii (in cazul nostru au fost 12), valoarea optima a functiei obiectiv pentru problema primala si duala, timpul CPU total in secunde, codul de terminare (0) si valoarea relativa a diferentei duale si asa mai departe.

```
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +8.45423e-09
```

In cele din urma starea problemei este afisata, in cazul de fata fiind rezolvata. Gasiti la adresa aceasta `http://cvxr.com/cvx/doc/solver.html#interpreting-the-results` interpretarea statusului in detaliu pentru restul starilor (de exemplu: Failed, Unbounded, etc). De asemenea se evidentiaza valoarea optima, aceasta regasindu-se si in rezumat.

In final daca nu se doreste afisarea acestor informatii, ele se pot suprima cu ajutorul cuvantului cheie **quiet**: `cvx_begin quiet ... cvx_end`

Stabilitatea sistemelor

Fie un sistem liniar dinamic $\dot{x} = Ax$. Dorim sa investigam daca el este stabil. Pentru aceasta trebuie sa verificam daca exista matricea X simetrica a.i.: $A^T X + XA \prec 0, \quad X \succ 0$

Cele 2 inegalitati stricte sunt omogene in X, deci problema poate fi formulata echivalent ca si:

$$A^T X + XA + I_n \preccurlyeq 0, \quad X \succcurlyeq I_n.$$

Deci obtinem o problema de programare semidefinita convexa [2] (*eng. semidefinite programming (SDP)*).

Amintim forma standard:

$$(SDP): \qquad \min_{x \in \mathbb{R}^n} \; c^T x, \quad \text{s.l.:} \quad LMI(x) \preccurlyeq 0, \; Ax - b = 0,$$

Putem rezolva aceasta problema in CVX, dupa cum urmeaza:

```
% A-eigenvalues uniform logarithmic spaced [-10^-1;-10^1]
n = 10;A=diag(-logspace(-0.5,1,n));U=orth(randn(n,n));
A=U'*A*U;
cvx_begin sdp
variable X(n,n) symmetric %Obs: diagonal,...
minimize(trace(X)) %Obs: poate lipsi aceasta linie
A'*X + X*A + eye(n) <= 0,
X >= eye(n)
cvx_end
```

2 Gradientul proiectat si gradientul conditional

Fie problema cu constrangeri:

$$\min_{x \in \Omega} f(x), \tag{1}$$

unde Ω este o multime simpla, convexa si compacta (inchisa si marginita).

O problema importanta din optimizarea constransa este problema proiectiei pe spatiul Euclidian:

[2] probleme convexe unde multimea fezabila este descrisa de LMI-uri

$$\min_{x\in\Omega} \|x - x_0\|^2$$

Proiectia Euclidiana a punctului x_0 pe multimea Ω: $[x_0]_\Omega \rightarrow$ determinarea celui mai "apropiat" punct (in distanta Euclidiana) din multimea Ω de punctul x_0.

Observatie: Proiectia exista si este unica daca multimea Ω este o multime convexa.

2.1 Metoda gradient proiectat

Pentru problema generala 1, vom deriva regula de updatare a metodei gradient proiectat pornind de la subproblema care a generat regula:

$$x_{k+1} = \arg\min_{x\in\Omega} f(x_k) + \nabla^T f(x_k)(x - x_k) + \frac{1}{2\alpha}\|x - x_k\|^2$$

Observam ca am putea forma un patrat perfect cat sa construim problema de proiectie Euclidiana:

$$x_{k+1} = \arg\min_{x\in\Omega} f(x_k) + \frac{1}{2\alpha}\left[\alpha^2\nabla^T f(x_k)\nabla f(x_k) + 2\alpha\nabla^T f(x_k)(x - x_k) + (x - x_k)^T(x - x_k)\right]$$
$$x_{k+1} = \arg\min_{x\in\Omega} f(x_k) + \frac{1}{2\alpha}\|x - (x_k - \alpha\nabla f(x_k))\|^2$$

Pseudocodul metodei gradient proiectat:

Algoritmul GP

Date de intrare : $\boldsymbol{x}_0\,(k = 0)$ punctul initial, pasul $\alpha_k > 0$

1. Atata timp cat `criteriu(`x_k`)` $\geq \epsilon$:

   1.1 $\boldsymbol{x}_{k+1} = [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_\Omega$

   1.2 $k = k + 1$

2.Returneaza x_{k+1}

Observatie: Spre deosebire de metoda gradient, metoda gradient proiectat are in plus un pas de proiectare pe multimea Ω, notat prin $[\cdot]_\Omega$. De retinut, ca metoda gradient proiectat se aplica cand proiectia este usor de calculat!

2.2 Metoda Gradient Conditional

Metoda gradient conditional se mai numeste si metoda Frank-Wolfe. Metoda aproximeaza functia obiectiv cu Taylor-ul de ordinul 1 in jurul punctului curent x_k sub aceasi constrangere:

$$s_k = \arg\min_{s\in\Omega} f(x_k) + \nabla^T f(x_k)(s - x_k) \Leftrightarrow$$
$$s_k = \arg\min_{s\in\Omega} \nabla^T f(x_k)(s - x_k),$$

unde am renuntat la termenul $f(x_k)$ ce nu depinde de s. Observati ca acum problema este una lineara in s si deci convexa. Metoda gradient conditional se utilizeaza cand complexitatea per iteratie este mult mai scazuta decat cea necesara pentru rezolvarea problemei originale. Pseudocodul este urmatorul:

Algoritmul GC

Date de intrare : $\boldsymbol{x}_0\,(k=0)$ punctul initial, pasul $\alpha_k > 0$

1. Atata timp cat `criteriu(`x_k`)` $\geq \epsilon$:

   1.1 $s_k = \arg\min_{s\in\Omega} \nabla^T f(x_k)(s - x_k)$

   1.2 $x_{k+1} = x_k + \alpha(s_k - x_k)$

   1.3 $k = k + 1$

2.Returneaza x_{k+1}

3 Matrix completion

Matrix completion are ca obiectiv completarea intrările lipsă dintr-o matrice parțial observată. Astfel:
Date de intrare: Se da o matrice $A \in \mathbb{R}^{n\times n}$ cu elemente lipsa.
Presupunere: A are rang scazut.
Scop: Gasirea elementelor lipsa.
Formularea problemei: In continuare prezentam doua formulari ce sunt neconvexe si foarte greu de abordat, dificultatea fiind data de gasirea rangului.

$$(P1): \min_{X\in\mathbb{R}^{n\times n}} \operatorname{rang}(X) \quad \overbrace{\text{sau}}^{\leftrightarrow} \quad (P2): \min_{X\in\mathbb{R}^{n\times n}} \sum_{i,j\in\Omega} \|X_{ij} - A_{ij}\|^2$$

$$\text{s.l.: } X_{ij} = A_{ij} \forall i,j \in \Omega \qquad\qquad \text{s.l.: } \operatorname{rang}(X) \leq r$$

unde Ω este multimea indicilor elementelor observabile din A.
Pentru **problema (P1)**, avem urmatoarea relaxare convexa:

$$(P1): \min_{X\in\mathbb{R}^{n\times n}} \operatorname{rang}(X) \quad \overbrace{\text{relaxare convexa}}^{\Rightarrow} \quad \min_{X\in\mathbb{R}^{n\times n}} \|X\|_*$$

$$\text{s.l.: } X_{ij} = A_{ij} \forall i,j \in \Omega \qquad\qquad \text{s.l.: } X_{ij} = A_{ij} \quad \forall i,j \in \Omega$$

unde $\|\cdot\|_*$ este norma nucleara si are urmatoarea definitie:

$$\|A\|_* = \sum_{i=1}^{r} \sigma_i, \quad \text{unde } A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top \,(DVS)$$

Gradientul proiectat(GP): Pentru a calcula (sub)gradientul functiei $f(X) = \|X\|_*$ se foloseste DVS-ul lui $X = U\Sigma V^T$:

$$\nabla f(X) = UV^T$$

Deci putem implementa urmatoarea iteratie (e.g. $c = 10$ si $X_0 = 0$ si $X_0(\Omega) = A(\Omega)$) :

$$X_{k+1} = \left[X_k - \alpha_k \nabla f\left(X_k\right)\right]_\Omega, \quad \alpha_k = \frac{c}{k}$$

Altfel spus:

$$\begin{aligned} X_k &= U_k \Sigma_k V_k^T \\ \bar{X}_k &= X_k - \frac{c}{k} U_k V_k^T \\ X_{k+1} &= \left[\bar{X}_k\right]_\Omega \end{aligned}$$

Pentru aplicatia de recuperare a imaginii (versiunea P1), proiectia este:

$$X_{k+1} = \bar{X}_k[i,j] = A[i,j] \quad \forall i,j \in \Omega$$

Pentru **problema (P2)**, avem urmatoarea relxarea convexa:

$$(P2): \min_{X \in \mathbb{R}^{n\times n},\ \text{rang}\ (X) \le r} \|\mathcal{P}(X - A)\|^2 \overbrace{\text{relaxare}}^{\text{convexa}} \qquad \min_{X \in \mathbb{R}^{n\times n}, \|X\|_* \le \delta_r} \|\mathcal{P}(X - A)\|^2$$

unde $\mathcal{P}$ operator liniar de proiectie pe componente. Utilizand urmatorul rezultatul matriceal:

$$\|X\|_* \le \delta \Leftrightarrow \exists W_1, W_2 : \operatorname{tr}(W_1) + \operatorname{tr}(W_2) \le 2\delta \ \& \ \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix} \succeq 0$$

obtinem o relaxare de tip SDP pentru $(P2)$:

$$\begin{aligned} (P2 - sdp) : \min_{X, W_1, W_2} & \sum_{i,j\in\Omega} \left(X_{ij} - A_{ij}\right)^2 \\ \text{s.l.:} \ & \operatorname{tr}(W_1) + \operatorname{tr}(W_2) \le 2\delta_r, \quad \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix} \succeq 0 \end{aligned}$$

Gradientul conditional (GC):

- calculul gradientlui se poate face eficient daca $\#\Omega$ este mica

$$\nabla f(X) = \mathcal{P}(X - A)$$

- subproblema de la fiecare pas se poate rezolva explicit:

$$S_k = \arg\min_X \left\langle \nabla f\left(X_k\right), X\right\rangle \ \text{ s.l.: } \|X\|_* \le \delta_r,$$

  considerand cea mai mare valoare singulara σ_1 a lui $\Delta = \nabla f\left(X_k\right)$ cu vectorii singulari u_1 (stang) si v_1 (drept):

$$S_k = -\delta_r u_1 v_1^T.$$

- obtinem o iteratie de forma (updatari de rang = 1)

$$X_{k+1} = (1 - \alpha_k) X_k + \alpha_k S_k, \quad \alpha_k = 2/(k+2)$$

Cerinta:

- Implementati problema (P1) versiunea convexa folosind CVX. (Hint: folositi *norm_nuc* pentru norma nucleara)
- Implementati gradientul proiectat pentru $\epsilon = 1e^{-3}$, $c = 10$ si utilizati criteriul de oprire $\|X_{k+1} - X_k\|_F$.
- Implementati Gradientul Conditional pentru $\delta_r = 70$ si utilizati criteriul de oprire $\|X_{k+1} - X_k\|_F$. Implementati metoda puterii pentru a gasi S_k
- Plotati folosind *semilogy* criteriul de oprire atat pentru GP cat si GC pe acelasi grafic.
- Afisati imaginile recuperate.

References

[1] Ion Necoara, Slide-ri curs Optimizari.

[2] `http://web.cvxr.com/cvx/doc/index.html`

[3] `http://sedumi.ie.lehigh.edu/`

[4] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3

[5] `https://www.gurobi.com/`

[6] `https://www.mosek.com/`