Ricerca Operativa - Laboratorio

Lezione 7 - Esempio di Programmazione Non Lineare: applicazione del metodo di Frank-Wolfe

Docente: Luigi De Giovanni

Dipartimento di Matematica “Tullio Levi-Civita”
Università degli Studi di Padova

luigi@math.unipd.it
https://www.math.unipd.it/∼luigi/

Corso di Laurea Magistrale in Matematica
Università degli Studi di Padova
a.a. 2019–2020

Ricostruzione sparsa di segnali: formulazione esatta

Dati:

- un segnale reale $b \in \mathbb{R}^m$ (segnale target);
- un dizionario di n segnali standard (spikes) $A = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$ con $m << n$;

Ricostruzione sparsa di segnali: formulazione esatta

Dati:

- un segnale reale $b \in \mathbb{R}^m$ (segnale target);
- un dizionario di n segnali standard (spikes) $A = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$ con $m << n$;

Obiettivo: rappresentare b come combinazione lineare del minor numero possibile di elementi del dizionario (spikes).

Ricostruzione sparsa di segnali: formulazione esatta

Dati:

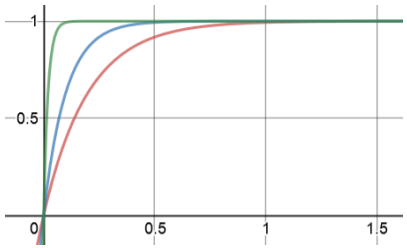
- un segnale reale $b \in \mathbb{R}^m$ (segnale target);
- un dizionario di n segnali standard (spikes) $A = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$ con $m << n$;

Obiettivo: rappresentare b come combinazione lineare del minor numero possibile di elementi del dizionario (spikes).

$$\begin{aligned} \min \ & \|x\|_0 \\ & Ax = b \\ & x \in \mathbb{R}^n \end{aligned}$$

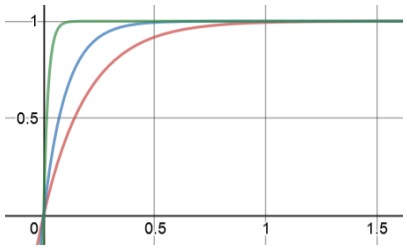
Ricostruzione sparsa di segnali: riformulazione

- $\|x\|_0 = \sum_{i=1}^{n} \theta(|x_i|)$, con $\theta(w) = \begin{cases} 0 \text{ se } w \leq 0 \\ 1 \text{ se } w > 0 \end{cases}$, $w \in \mathbb{R}$
- se $w \geq 0$, $\theta(w) \cong 1 - e^{r\,|w|}$, con $r > 0$
  (un tipico valore è $r = 5$)
- trasformazione $y_i = |x_i|$



Ricostruzione sparsa di segnali: riformulazione

- $\|x\|_0 = \sum_{i=1}^{n} \theta(|x_i|)$, con $\theta(w) = \begin{cases} 0 \text{ se } w \leq 0 \\ 1 \text{ se } w > 0 \end{cases}$, $w \in \mathbb{R}$
- se $w \geq 0$, $\theta(w) \cong 1 - e^{r|w|}$, con $r > 0$
  (un tipico valore è $r = 5$)
- trasformazione $y_i = |x_i|$



$$\min \sum_{i=1}^{n} \left(1 - e^{-r y_i}\right)$$
$$Ax = b$$
$$-y_i \leq x_i \leq y_i, \quad i = 1, \ldots, n$$

Scelta del metodo di soluzione

- Problema con funzione obiettivo concava e vincoli lineari

Scelta del metodo di soluzione

- Problema con funzione obiettivo concava e vincoli lineari
- È possibile utilizzare Frank-Wolfe con passo unitario

Scelta del metodo di soluzione

- Problema con funzione obiettivo concava e vincoli lineari
- È possibile utilizzare Frank-Wolfe con passo unitario
- Si dovranno risolvere un numero finito di problemi di PL

Scelta del metodo di soluzione

- Problema con funzione obiettivo concava e vincoli lineari
- È possibile utilizzare Frank-Wolfe con passo unitario
- Si dovranno risolvere un numero finito di problemi di PL
- Utilizzare un algoritmo per problemi di PL che produce vertici ottimi (per esempio CPLEX)

Metodo di Frank-Wolfe

Scegli un punto di partenza $x^0 \in \Omega$

For $k = 0, 1, \ldots$

Calcola $\bar{x}^k \in \text{Argmin}_{x \in \Omega}\{\nabla f(x^k)^T x\}$

If $\nabla f(x^k)^T(\bar{x}^k - x^k) \geq 0$, STOP

Else

Poni $d^k = \bar{x}^k - x^k$

Scegli $\alpha^k \in (0, 1]$ attraverso una line search

Poni $x^{k+1} = x^k + \alpha^k d^k$

End if

End For

Aspetti implementativi

- $\alpha^k = 1$, per ogni k (passo unitario: min funzione concava su politopo)

Aspetti implementativi

- $\alpha^k = 1$, per ogni k (passo unitario: min funzione concava su politopo)
- scelta punto iniziale: risolvere un problema di PL opportuno (per esempio con $e^T x$ per funzione obiettivo)

Aspetti implementativi

- $\alpha^k = 1$, per ogni k (passo unitario: min funzione concava su politopo)
- scelta punto iniziale: risolvere un problema di PL opportuno (per esempio con $e^T x$ per funzione obiettivo)
- criterio di arresto: $\nabla f(x^k)^T(\bar{x}^k - x^k) \geq -\epsilon$ per ϵ "sufficientemente piccolo"

Generazione istanze

- dimensioni $m = 32$ e $n = 128$
- A matrice randomica (Normal$(0, 1)$)
- $b = Ax^\star + noise$
- $x^\star$ soluzione generata randomicamente con (al più) 5 componenti non nulle (sapendo con quali spike è stato generato il segnale target, potremmo verificare che la soluzione prodotta dal metodo proposto corrisponda a $x^\star$)
- *noise* generato randomicamente come Normal$(0, 0.001)$

Implementazione in AMPL

- `fw.mod`: contiene la definizione dei parametri utili alla generazione delle istanze, e il modello con funzione obiettivo linearizzata da risolvere a ogni iterazione
- `fw.dat`: contiene i valori dei parametri per la generazione casuale delle istanze e per la riformulazione del modello
- `fw.run`: contiene lo script che implementa il metodo di Frank-Wolfe e visualizza i risultati ottenuti

Frank-Wolfe per ricostruzione sparsa di segnali (i)

fw.mod

```
#DEFINIZIONE DEL PROBLEMA
param n; # numero colonne di A
param m; # numero righe di A
set RIGHE := 1..m;
set COLONNE := 1..n;
param A{RIGHE,COLONNE} := Normal(0,1);

#SEGNALE TARGET b GENERATO CASUALMENTE su base di l spikes
param l; # numero componenti non nulle nel segnale target
set SPIKES := 1..l;
param ind_spikes{SPIKES} := floor(Uniform(1,n+1));
param sign_spikes{SPIKES} := Uniform(-10,10);
param b{i in RIGHE} :=
  sum {k in SPIKES} A[i,ind_spikes[k]]*sign_spikes[k]
  + Normal(0,0.001);
```

Frank-Wolfe per ricostruzione sparsa di segnali (ii)

```
#PROBLEMA LINEARIZZATO PER ITERAZIONE FRANK-WOLFE
param r; # parametro per approssimare la norma zero
param c_k{COLONNE}; # coefficienti f.o. linearizzata
    #  (gradiente nel punto corrente)

#nuovo punto per direzione ammissibile
var x_k{COLONNE};
var y_k{COLONNE};

# f.o. linearizzata (con il gradiente nel punto corrente)
minimize fw_obj: sum{j in COLONNE} c_k[j]*y_k[j];
# vincoli del problema
s.t. v_riproduci_target{i in RIGHE} :
   sum {j in COLONNE} A[i,j]*x_k[j] = b[i];
s.t. v_abs_l{j in COLONNE}: y_k[j] >= -x_k[j];
s.t. v_abs_u{j in COLONNE}: y_k[j] >=  x_k[j];
```

Frank-Wolfe per ricostruzione sparsa di segnali (iii)

fw.dat

```
#dati del problema
param m := 32;
param n := 128;
param l := 5;

#parametri formulazione
param r := 5;
```

Frank-Wolfe per ricostruzione sparsa di segnali (iv)

fw.run

```
reset;
option randseed 1;
model fw.mod;
data fw.dat;

printf "\n ======================\n";
printf   " METODO DI FRANK-WOLFE\n";
printf   " ======================\n";

option solver cplexamp;

param x_curr{COLONNE}; # soluzione corrente
param y_curr{COLONNE}; # soluzione corrente
param eps := 1e-6; # condizione di uscita dall'algoritmo
param it; # contatore iterazioni

# inizializza x_curr e y_curr
for {j in 1..n} {
  let x_curr[j] := 1.0;
  let y_curr[j] := abs(x_curr[j]);
}
```

Frank-Wolfe per ricostruzione sparsa di segnali (v)

```
let it := 0;
repeat {
  printf "\n* ITERAZIONE %d\n", it;
  if (it >1) then {
    # passa alla nuova soluzione corrente
    for {j in 1..n} {
      let x_curr[j] := x_k[j];
      let y_curr[j] := y_k[j];
    }
  }

  # calcolo dei coefficienti della f.o. linearizzata
  for {j in 1..n} {
    let c_k[j] := r*exp(-r*y_curr[j]);
  }

  solve;

  let it := it + 1;
} until (it>1 && (sum {j in COLONNE} c_k[j]*(y_k[j]-y_curr[j]) >= -eps));
```

Frank-Wolfe per ricostruzione sparsa di segnali (vi)

```
printf "\n================================================\n";
param soglia_visualizzazione = 0.001;
printf "Soluzione trovata (componenti > %f):\n", soglia_visualizzazione;
param ricostruito;
param effettivo;
for {j in COLONNE} {
  let effettivo := 0;
  for { k in SPIKES } {
    if ( j == ind_spikes[k] ) then {
      let effettivo := sign_spikes[k];
    }
  }
  let ricostruito := x_k[j];
  if ( abs(effettivo) > soglia_visualizzazione
          or abs(ricostruito) > soglia_visualizzazione ) then {
    printf "spike # x_k [%3d] = %+12.6f # %+12.6f\n", j,
      effettivo, ricostruito;
  }
}
printf "\n================================================\n";
printf "Totale componenti: spike # x_k = %d # %d\n",
  card(SPIKES),
  card({j in COLONNE:abs(x_k[j])>soglia_visualizzazione});
```