Clustering e metodi di decomposizione

Marco Sciandrone
Dipartimento di Ingegneria dell'Informazione
Università di Firenze
E-mail: marco.sciandrone@unifi.it

Sommario

In questi appunti viene considerato il problema di clustering, ossia il problema di partizionare un insieme di oggetti (*pattern*) in gruppi "omogenei" (*cluster*). Verrà presentata una formulazione del problema di programmazione matematica mista e verrà mostrata l'equivalenza con un problema di ottimizzazione continua. Verrà inoltre descritto il metodo *k-means*, sia nella versione *batch* che in quella *incrementale*, analizzato come metodo di decomposizione.

1 Il problema del clustering

Il problema del *clustering* è un problema centrale nell'ambito dell'*apprendimento automatico non supervisionato* e, in termini molto generali, può essere sintetizzato nel modo seguente: dato un insieme di *pattern*, determinare una partizione in sottoinsiemi *cluster* che siano *omogenei* e/o ben separati.

In queste note assumeremo che il numero di cluster è fissato, e definiremo una funzione obiettivo (connessa all'omogeneità e/o alla separazione dei cluster) che deve essere ottimizzata.

Esempio di cluster analysis

Si supponga che il comportamento dei clienti di una compagnia telefonica sia analizzato sulla base di due caratteristiche:

- *attenzione ai costi*;
- *fedeltà*.

Ognuna delle due caratteristiche può essere rappresentata da una variabile che assume valori discreti (ad esempio, 1,. . .,9), tali per cui il valore più alto indica, rispettivamente, massima attenzione ai costi e massima fedeltà. Ogni cliente è un pattern (vettore) bidimensionale, le cui componenti sono state determinate sulla base di un'analisi del comportamento *storico* del cliente. Si vogliono raggruppare i clienti in M insiemi che siano "simili" rispetto alle due caratteristiche considerate, al fine di tenere conto, per eventuali operazioni commerciali, delle differenti "tipologie" di cliente. □

Il problema del clustering, ossia il problema di partizionare un insieme di vettori appartenenti a R^n in un numero prefissato di cluster, può essere formulato in modi differenti, e per ogni formulazione esistono specifici algoritmi.

Considereremo una tra le più adottate formulazioni.

Sia $d(.,.) : R^n \times R^n \to R^+$ una misura di *dissimilarità* tra due vettori in R^n. Il concetto di dissimilarità è generale. Introdotta una norma $\|\cdot\|$ in R^n, comunque presi due vettori $x, y \in R^n$, si può pensare di definire la dissimilarità $d(x, y)$ tra x e y come la *distanza*

$$d(x, y) = \|x - y\|.$$

Definiamo la *dissimilarità* di un cluster come la somma delle *dissimilarità* tra i vettori del cluster e un *centroide* (che deve essere determinato) del cluster.

Sia dato l'insieme (training set) dei pattern

$$TS = \{x^i,\ x^i \in X \subseteq R^n,\ i = 1, \ldots, N\},$$

e sia M il numero di cluster assegnato.

Si vuole partizionare il training set in modo da minimizzare la somma delle *dissimilarità* degli M cluster.

Al fine di definire una formulazione del problema in termini di programmazione matematica, introduciamo le *variabili binarie*

$$\delta_{ij} = \begin{cases} 1 & \text{se } x^i \in \text{cluster } j \\ 0 & \text{altrimenti} \end{cases}$$

e i vettori di *variabili continue* $z_j \in R^n$, $j = 1, \ldots, M$, ognuno dei quali rappresenta un centroide associato al corrispondente cluster. Per definizione, la dissimilarità del cluster j è

$$s_j = \sum_{i=1}^{N} \delta_{ij} d(x^i, z_j).$$

Ogni pattern x^i, per $i = 1, \ldots, N$ deve essere assegnato ad uno ed uno solo cluster, per cui si hanno i vincoli

$$\sum_{j=1}^{M} \delta_{ij} = 1 \qquad i = 1, \ldots, N.$$

Di conseguenza il problema assume la forma di problema di *programmazione matematica mista*

$$\begin{aligned} & \min_{\delta, z} \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{ij} d(x^i, z_j) \\ & \sum_{j=1}^{M} \delta_{ij} = 1 \qquad i = 1, \ldots, N \\ & \delta_{ij} \in \{0, 1\} \qquad i = 1, \ldots, N, j = 1, \ldots, M \\ & z_j \in R^n \qquad j = 1, \ldots, M \end{aligned} \tag{1}$$

Possiamo dimostrare (si veda l'Appendice A) l'equivalenza del problema (1) con il seguente problema di *programmazione matematica continua*

$$
\begin{aligned}
&\min_{\delta,z} \sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}d(x^i,z_j) \\
&\sum_{j=1}^{M}\delta_{ij} = 1 \qquad i=1,\ldots,N \\
&\delta_{ij} \geq 0 \qquad i=1,\ldots,N, j=1,\ldots,M \\
&z_j \in R^n \qquad j=1,\ldots,M
\end{aligned}
\tag{2}
$$

Si osservi che il problema (2) è un *rilassamento continuo* del problema (1).

Per comodità di esposizione, in alcuni casi faremo riferimento al problema (2) con la notazione

$$
\begin{aligned}
&\min_{\delta,z} f(\delta,z) = \sum_{i=1}^{N}\delta_i^T d_i(z) \\
&e^T\delta_i = 1 \qquad i=1,\ldots,N \\
&\delta_i \in R^M \qquad \delta_i \geq 0 \qquad i=1,\ldots,N, \\
&z_j \in R^n \qquad j=1,\ldots,M,
\end{aligned}
\tag{3}
$$

dove $\delta_i \in R^M$, e è il vettore con tutte le componenti pari a 1,

$$
d_i(z) = \begin{pmatrix} d(x^i,z_1) \\ d(x^i,z_2) \\ \vdots \\ d(x^i,z_M) \end{pmatrix}
$$

Fissato $\bar{z}$, il problema (3), rispetto a δ, è decomponibile in N problemi separabili. In particolare, l'$i-$esimo problema assume la forma

$$
\begin{aligned}
&\min_{\delta_i} f_i(\delta_i,\bar{z}) = d_i(\bar{z})^T\delta_i \\
&e^T\delta_i = 1 \\
&\delta_i \geq 0,
\end{aligned}
\tag{4}
$$

ossia è un problema con funzione obiettivo *lineare* e vincoli di tipo *simplesso*. Una soluzione $\delta_i^\star$ di (4) può essere ottenuta come segue (si veda l'Appendice B):

- sia $j^\star$ tale che

$$
d_{ij^\star}(\bar{z}) \leq d_{ij}(\bar{z}) \qquad j=1,\ldots,M;
$$

- si pone

$$\delta_{ij}^{\star} = \begin{cases} 1 & \text{se } j = j^{\star} \\ \\ 0 & \text{altrimenti} \end{cases}$$

2 Algoritmi di tipo $k-$means

Spesso le tecniche di clustering si basano sull'impiego della norma euclidea per definire la funzione di dissimilarità. Ponendo $d(x,z) = \frac{1}{2}\|x-z\|^2$, il problema (2) diviene

$$\begin{aligned} &\min_{\delta,z} f(\delta,z) = \tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\|x^i - z_j\|^2 \\ \\ &\sum_{j=1}^{M}\delta_{ij} = 1 \qquad i = 1,\ldots,N \\ \\ &\delta_{ij} \geq 0 \qquad i = 1,\ldots,N, j = 1,\ldots,M \\ \\ &z_j \in R^n \qquad j = 1,\ldots,M \end{aligned} \tag{5}$$

L'algoritmo più noto per la soluzione di (5) è l'algoritmo *k-means*. Considereremo sia le versione *batch* che quella *incrementale* dell'algoritmo (si veda l'Appendice C per le definizioni di algoritmo di tipo batch e di tipo incrementale).

Algoritmo $k-$means di tipo batch

In questa versione dell'algoritmo viene effettuata una *minimizzazione alternata* di $f(\delta,z)$ rispetto ai due blocchi di variabili δ e z. L'algoritmo $k-$means nella versione batch puòquindi essere visto come un algoritmo di *decomposizione in due blocchi di tipo Gauss-Seidel*. La struttura della funzione obiettivo e i vincoli su δ implicano che il calcolo del minimo globale rispetto ad ogni blocco può essere effettuato in forma chiusa.

Fissato $\bar{z}$, il problema di minimo rispetto a δ assume la forma (4), ossia è un problema di programmazione lineare decomponibile in N problemi separabili, ognuno dei quali può essere risolto per via analitica come mostrato nel paragrafo precedente.

Fissato $\bar{\delta}$, il problema di minimo rispetto a z è decomponibile in M problemi quadratici convessi, dove il $j-$esimo problema assume la forma

$$\begin{aligned} &\min_{z_j} f_j(z_j) = \tfrac{1}{2}\sum_{i=1}^{N}\bar{\delta}_{ij}\|x^i - z_j\|^2 \\ \\ &z_j \in R^n, \end{aligned} \tag{6}$$

la cui soluzione $z_j^{\star}$ si ottiene uguagliando a zero il gradiente

$$\nabla_{z_j} f(\bar{\delta}, z_j) = \sum_{i=1}^{N}\bar{\delta}_{ij}(z_j - x^i),$$

ossia, assumendo $\sum_{i=1}^{N} \bar{\delta}_{ij} \neq 0$, si pone

$$z_j^\star = \frac{\sum_{i=1}^{N} \bar{\delta}_{ij} x^i}{\sum_{i=1}^{N} \bar{\delta}_{ij}}$$

L'algoritmo è formalmente descritto come segue.

Algoritmo k-means: versione batch

Dati. $z^0 \in R^{nM}$. $k = 0$

While il criterio d'arresto non è soddisfatto **do**

Minimizzazione vincolata rispetto a δ

Per $i = 1, \ldots, N$ calcola *la soluzione binaria*

$$\delta_i^{k+1} \in \arg\min\{f_i(\delta_i, z^k) : \sum_{j=1}^{M} \delta_{ij} = 1,\ \delta_{ij} \geq 0\},$$

i.e., sia $j(k) \in \arg\min_{j=1,\ldots,M}\{\|x^i - z_j\|\}$ e poni

$$\delta_{ij}^{k+1} = \begin{cases} 1 & \text{se } j = j(k) \\ 0 & \text{altrimenti,} \end{cases}$$

Minimizzazione non vincolata rispetto a z

Calcola $z^{k+1} \in \arg\min_z f(\delta^{k+1}, z)$, i.e., per $j = 1, \ldots, M$,
se $\nabla_{z_j} f(\delta^{k+1}, z^k) = 0$ poni $z_j^{k+1} = z_j^k$,
altrimenti poni

$$z_j^{k+1} = \frac{\sum_{i=1}^{N} \delta_{ij}^{k+1} x^i}{\sum_{i=1}^{N} \delta_{ij}^{k+1}}.$$

$k = k + 1$.

End While

Un criterio d'arresto ragionevole potrebbe essere il seguente

$$\frac{|f(\delta^{k+1}, z^{k+1}) - f(\delta^k, z^k)|}{1 + f(\delta^k, z^k)} \leq \epsilon \tag{7}$$

con $\epsilon > 0$.

Se si pone $\epsilon = 0$ in (7), tenendo conto che il numero di vettori δ visitati dall'algoritmo è finito, si può dimostrare che l'algoritmo *termina in un numero finito di passi* in un *punto critico* $(\delta^\star, z^\star)$ che gode di ulteriori proprietà, ossia

$$f(\delta^\star, z^\star) \leq f(\delta, z^\star) \quad \forall \text{ vettore ammissibile } \delta$$

$$f(\delta^\star, z^\star) \leq f(\delta^\star, z) \quad \forall z$$

Proposizione 1 *Si assuma nell'Algoritmo* $k-$means versione batch *sia adottato il criterio d'arresto* (7) *con* $\epsilon = 0$. *Allora l'algoritmo termina in un numero finito di iterazioni in un punto critico* $(\delta^\star, z^\star)$ *tale che*

$$\begin{aligned} &f(\delta^\star, z^\star) \leq f(\delta, z^\star) \quad \forall \text{ vettore ammissibile } \delta \\ &f(\delta^\star, z^\star) \leq f(\delta^\star, z) \quad \forall z \end{aligned} \tag{8}$$

Dim. Si supponga, per contraddizione, che l'algoritmo generi una sequenza infinita. Di conseguenza si ha per ogni k

$$f(\delta^{k+1}, z^{k+1}) < f(\delta^k, z^k). \tag{9}$$

Le istruzioni dell'algoritmo implicano, per ogni $k > 0$,

$$f(\delta^k, z^k) \leq f(\delta^k, z) \qquad \forall z \in R^{nM}, \tag{10}$$

$$f(\delta^{k+1}, z^k) \leq f(\delta, z^k) \qquad \forall \text{ vettore ammissibile } \delta. \tag{11}$$

Poiché δ^k è un vettore a componenti binarie, esso appartiene a un insieme finito di punti, e quindi possiamo estrarre un sottoinsieme infinito $K \subseteq \{0, 1, \ldots, \}$ tale che

$$\delta^k = \bar{\delta} \qquad \forall k \in K.$$

Per ogni $k \in K$ sia $l(k) \geq 1$ tale che $k + l(k) \in K$, per cui possiamo scrivere

$$f(\delta^k, z^k) = f(\bar{\delta}, z^k) \qquad f(\delta^{k+l(k)}, z^{k+l(k)}) = f(\bar{\delta}, z^{k+l(k)}).$$

Dalla (10) e dalla (9) segue

$$f(\bar{\delta}, z^k) \leq f(\bar{\delta}, z^{k+l(k)}) < f(\bar{\delta}, z^k),$$

ossia una contraddizione.

Abbiamo quindi dimostrato che l'algoritmo termina in un numero finito di iterazioni in un punto $(\delta^\star, z^\star) = (\delta^k, z^k)$ tale che

$$f(\delta^{k+1}, z^{k+1}) = f(\delta^{k+1}, z^k) = f(\delta^k, z^k) = f(\delta^\star, z^\star). \tag{12}$$

Dalla (10) segue

$$f(\delta^\star, z^\star) \leq f(\delta^\star, z) \qquad \forall z.$$

La (11) e la (12) implicano

$$f(\delta^\star, z^\star) = f(\delta^{k+1}, z^k) \leq f(\delta, z^k) = f(\delta, z^\star) \qquad \forall \text{ vettore ammissibile } \delta$$

e quindi la (8) è dimostrata.

Infine, la (8) e la continua differenziabilità di f implicano che il punto $(\delta^\star, z^\star)$ soddisfa le condizioni necessarie del primo ordine di ottimalità e quindi è un punto critico. □

Algoritmo $k-$means di tipo incrementale

La versione incrementale dell'algoritmo k-means può essere descritto in modo informale come segue

- Scegli in modo random i vettori z_j^0 e poni $k = 0$;
- Seleziona un pattern $x(k)$ dal training set;
- sia $j(x)$ il *best-matching (winning)* centroide per il pattern $x(k)$, ossia

$$j(x) \in \arg \min_{j=1,\ldots,M} \|x(k) - z_j^k\|;$$

- poni

$$z_j^{k+1} = \begin{cases} z_j^k + \alpha(x(k) - z_j^k) & \text{se } j = j(x) \\ z_j^k & \text{altrimenti} \end{cases} \tag{13}$$

L'idea alla base dell'algoritmo è di muovere i centroidi, distribuiti in modo casuale all'inizio, verso le regioni dello spazio a maggiore densità di pattern. Questo avviene mediante uno spostamento di ampiezza α lungo la direzione $x(k) - z_j^k$, dove il pattern $x(k)$ è selezionato in modo random (favorendo in tal modo le regioni dello spazio a maggiore densità), e il centroide z_j^k da muovere è quello a distanza minima da $x(k)$.

Faremo vedere che l'algoritmo corrisponde ad un algoritmo di ottimizzazione di tipo incrementale basato su una tecnica di decomposizione.

Osserviamo che la funzione obiettivo $f(\delta, z)$ del problema (5) può essere scritta come segue

$$f(\delta_1, \delta_2, \ldots, \delta_N, z_1, z_2 \ldots, z_M) = \sum_{i=1}^{N} f_i(\delta_i, z),$$

dove

$$f_i(\delta_i, z) = \frac{1}{2}\sum_{j=1}^{M}\delta_{ij}\|x^i - z_j\|^2,$$

e inoltre si ha

$$\nabla_{z_j} f_i(\delta_i, z) = -\delta_{ij}(x^i - z_j).$$

Mostreremo che alla generica iterazione k, l'algoritmo:

- seleziona un indice $i \in \{1, \ldots, N\}$,
- effettua una *minimizzazione esatta vincolata* di $f_i(\delta_i, z^k)$ rispetto a δ_i (tenendo fisso z^k) ottenendo δ_i^{k+1},
- aggiorna z^k con una strategia di tipo *gradiente incrementale*, ossia

$$z^{k+1} = z^k - \alpha^k \nabla_z f_i(\delta_i^{k+1}, z^k).$$

Quindi, come anticipato, l'algoritmo combina una strategia di tipo *incrementale* con una tecnica di *decomposizione in due blocchi*.

Si osservi che la minimizzazione esatta vincolata di $f_i(\delta_i, z^k)$ richiede il calcolo di una soluzione del problema lineare del tipo seguente

$$\begin{aligned} &\min_{\delta_i} f_i(\delta_i, z^k) = d_i^T \delta_i \\ &e^T \delta_i = 1 \\ &\delta_i \geq 0. \end{aligned} \tag{14}$$

dove si è omessa, per comodità di esposizione, la dipendenza di d_i da z^k, la cui $j-$esima componente è

$$d_{ij}(z^k) = 1/2\|x^i - z_j^k\|^2.$$

Per quanto visto in precedenza, una soluzione δ_i^{k+1} si calcola selezionando l'indice $j(k) \in \arg\min_{j=1,\ldots,M}\{\|x^i - z_j^k\|\}$ corrisponendente al *best-matching (winning)* centroide, e ponendo

$$\delta_{ij}^{k+1} = \begin{cases} 1 & \text{se } j = j(k) \\ \\ 0 & \text{altrimenti,} \end{cases}$$

La descrizione formale dell'algoritmo è riportata di seguito.

Algoritmo K-means: versione incrementale

Dati. $z^0 \in R^{nM}$. $k = 0$

While il criterio d'arresto non è soddisfatto **do**

Seleziona $i \in \{1, \ldots, N\}$ (**Ciclo incrementale**)

calcola calcola *la soluzione binaria*

$$\delta_i^{k+1} \in \arg\min\{f_i(\delta_i, z^k) : \sum_{j=1}^{M} \delta_{ij} = 1,\ \delta_{ij} \geq 0\},$$

i.e., sia $j(k) \in \arg\min_{j=1,\ldots,M}\{\|x^i - z_j\|\}$ e poni

$$\delta_{ij}^{k+1} = \begin{cases} 1 & \text{se } j = j(k) \\ 0 & \text{altrimenti,} \end{cases}$$

poni

$$z^{k+1} = z^k - \alpha^k \nabla_z f_i(\delta_i^{k+1}, z^k),$$

$k = k + 1$

End While

Per verificare la corrispondenza tra la versione informale e quella formale dell'algoritmo, ricordando l'espressione di δ_{ij}^{k+1}, possiamo scrivere

$$f_i(\delta_i^{k+1}, z) = 1/2\|x^i - z_{j(k)}\|^2,$$

e quindi si ha

$$\nabla_{z_j} f_i(\delta_i^{k+1}, z^k) = \begin{cases} -(x^i - z_j) & \text{se } j = j(k) \\ 0 & \text{altrimenti,} \end{cases}$$

da cui segue la regola di aggiornamento (13).

Anche in questo caso, un criterio d'arresto ragionevole potrebbe essere basato sulla variazione relativa della funzione obiettivo. Infine, non sono note prove di convergenza teorica della versione incrementale dell'algoritmo presentata.

3 Appendice A: equivalenza tra problemi di ottimizzazione

In molti casi può essere conveniente trasformare un problema assegnato in un altro problema, a esso "equivalente", nel senso che si può ottenere facilmente una soluzione del problema originario a partire da una soluzione del problema trasformato. In questo paragrafo ci limitiamo a fornire una breve giustificazione di trasformazioni elementari, che possono essere utilizzate, ad esempio, per eliminare non differenziabilità nella funzione obiettivo o per trasformare problemi vincolati (con vincoli semplici) in problemi non vincolati.

Consideriamo due problemi di ottimo del tipo:

$$\begin{array}{ll} min & f(x) \quad \text{(P)} \\ x \in S & \end{array} \qquad\qquad \begin{array}{ll} min & g(y) \quad \text{(Q)} \\ y \in D & \end{array}$$

dove il problema (P) ha insieme ammissibile $S \subseteq R^n$ e funzione obiettivo $f : R^n \to R$, il problema (Q) ha insieme ammissibile $D \subseteq R^m$ e funzione obiettivo $g : R^m \to R$.

I due problemi risultano *equivalenti* se:

- l'esistenza della soluzione di uno dei due problemi deve implicare e essere implicata dall'esistenza della soluzione dell'altro problema;
- è possibile stabilire una corrispondenza tra le soluzioni dei due problemi.

Un semplice criterio di equivalenza tra i due problemi è il seguente.

Proposizione 2 *Siano $S \subseteq R^n$, $f : S \to R$, $D \subseteq R^m$, $g : D \to R$ tali che:*

(i) *in corrispondenza a ciascun $x \in S$ è possibile determinare un $y \in D$ tale che $g(y) \leq f(x)$;*

(ii) *in corrispondenza a ciascun $y \in D$ è possibile determinare un un $x \in S$ tale che $f(x) \leq g(y)$.*

Allora f ha un punto di minimo globale su S se e solo se g ha un punto di minimo globale su D.

Dim. Supponiamo che g abbia un punto di minimo globale y^* su D. Per la (ii) esiste $x^\star \in S$ tale che $f(x^\star) \leq g(y^\star)$. Mostriamo che $x^\star$ è un punto di minimo globale di f su S. Supponiamo, per assurdo, che esista un $\bar{x} \in S$ tale che $f(\bar{x}) < f(x^\star)$. Dalla condizione (i) segue che esiste un punto $\bar{y} \in D$ tale che $g(\bar{y}) \leq f(\bar{x})$. Tenendo conto del fatto che $y^\star$ è un punto di minimo globale di g su D, e utilizzando le precedenti diseguaglianze si ottiene:

$$g(y^\star) \leq g(\bar{y}) \leq f(\bar{x}) < f(x^\star) \leq g(y^\star),$$

che è una contraddizione. L'implicazione inversa si dimostra in modo del tutto analogo, scambiando il ruolo dei due problemi. □

La proposizione precedente richiede che sia possibile definire due trasformazioni $\rho : S \to D$, $\sigma : D \to S$, tra gli insiemi ammissibili dei due problemi, in modo tale che a ogni $x \in S$ sia associato un elemento $\rho(x) \in D$ e, analogamente, a ogni $y \in D$ sia associato un elemento $\sigma(y) \in S$. Si richiede inoltre che siano verificate le condizioni:

Per ogni $x \in S$, l'elemento $\rho(x) \in Y$ soddisfa $g(\rho(x)) \leq f(x)$.

Per ogni $y \in D$, l'elemento $\sigma(y) \in S$ soddisfa: $f(\sigma(y)) \leq g(y)$.

Esempio 1.

Consideriamo il problema (P)

$$min \max_{1 \leq i \leq m} \{f_i(x)\}, \quad x \in S$$

in cui la funzione obiettivo

$$f(x) = \max_{1 \leq i \leq m} \{f_i(x)\}$$

è costituita dal massimo fra un numero finito di funzioni $f_i : S \to R$. Definiamo il nuovo problema (Q) nelle nuove variabili $z \in R$, $x \in S$

$$\begin{array}{ll} min & z \\ & f_i(x) \leq z, \quad i = 1, \ldots, m, \quad x \in S \end{array}$$

L'insieme ammissibile di (Q) è dato da

$$D = \left\{ y = \begin{pmatrix} z \\ x \end{pmatrix} : \quad f_i(x) \leq z, \quad i = 1, \ldots, m, \quad x \in S \right\}$$

e la funzione obiettivo di (Q) si può definire ponendo $g(y) = z$. Introduciamo ora le trasformazioni

$$\rho(x) = \begin{pmatrix} \max\limits_{1 \leq i \leq m} \{f_i(x)\} \\ x \end{pmatrix}, \qquad \sigma(y) = \sigma \begin{pmatrix} z \\ x \end{pmatrix} = x.$$

È immediato verificare che $\rho(x) \in D$ e che $\sigma(y) \in S$. Inoltre si ha:

$$g(\rho(x)) = \max_{1 \leq i \leq m} \{f_i(x)\} = f(x)$$

$$f(\sigma(y)) = f(x) = \max_{1 \leq i \leq m} \{f_i(x)\} \leq z = g(y),$$

per cui risultano soddisfatte le ipotesi della proposizione precedente.

Esempio 2.

Consideriamo il problema (P)

$$min_x \sum_{i=1}^{m} |f_i(x)|, \quad x \in S$$

Utilizzando la Proposizione 2 si può dimostrare che il problema precedente è equivalente al seguente problema

$$min_{x,z} \sum_{i=1}^{m} z_i$$

$$-z_i \leq f_i(x) \leq z_i \qquad i = 1, \ldots, m$$

$$x \in S.$$

La trasformazioni sono

$$\rho(x) = \begin{pmatrix} |f_1(x)| \\ |f_2(x)| \\ \vdots \\ |f_m(x)| \\ x \end{pmatrix}, \qquad \sigma(y) = \sigma \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \\ x \end{pmatrix} = x.$$

Esempio 3.

Consideriamo il problema (P)

$$min_x \sum_{i=1}^{m} F(|f_i(x)|), \quad x \in S,$$

dove $F : R^+ \to R$ è una funzione *monotona crescente*.

Utilizzando la Proposizione 2 si può dimostrare che il problema precedente è equivalente al seguente problema

$$min_{x,z} \sum_{i=1}^{m} F(z_i)$$

$$-z_i \leq f_i(x) \leq z_i \qquad i = 1, \ldots, m$$

$$x \in S.$$

Si osservi che l'ipotesi di funzione $F : R^+ \to R$ monotona crescente serve perché, posto

$$\sigma(y) = x,$$

si ha

$$f(\sigma(y)) \leq g(y),$$

essendo

$$|f_i(x)| \leq z_i.$$

3.1 Formulazione continua del problema di clustering

In questo paragrafo dimostreremo l'equivalenza tra i problemi seguenti (il primo di programmazione mista, il secondo di programmazione continua):

$$
\begin{aligned}
&\min_{\delta,z} \sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}d(x^i, z_j) \\
&\sum_{j=1}^{M}\delta_{ij} = 1 \qquad i = 1,\dots,N \\
&\delta_{ij} \in \{0,1\} \qquad i = 1,\dots,N, j = 1,\dots,M \\
&z_j \in R^n \qquad j = 1,\dots,M
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
&\min_{\delta,z} \sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}d(x^i, z_j) \\
&\sum_{j=1}^{M}\delta_{ij} = 1 \qquad i = 1,\dots,N \\
&\delta_{ij} \geq 0 \qquad i = 1,\dots,N, j = 1,\dots,M \\
&z_j \in R^n \qquad j = 1,\dots,M
\end{aligned}
\tag{16}
$$

Proposizione 3 (Equivalenza dei problemi)
Il problema (15) *di programmazione mista è equivalente al problema di programmazione continua* (16).

Dim. Per dimostrare la tesi utilizzeremo il risultato di equivalenza della Proposizione 2. A questo fine, il problema (P) è il problema (15), e il problema (Q) è il problema (16). Le variabili x e y dei problemi (P) e (Q) sono

$$x = \begin{pmatrix} z \\ \\ \delta \end{pmatrix} \qquad y = \begin{pmatrix} z \\ \\ \delta \end{pmatrix}.$$

Gli insiemi ammissibili S e D dei problemi (P) e (Q) sono tali che $S \subset D$ essendo il problema (Q) un *rilassamento continuo* del problema (P).

Per ogni $x \in S$ e $y \in S$, con $x = y$, si ha

$$f(x) = g(y). \tag{17}$$

La funzione $\rho : S \to D$ definita come segue

$$\rho(x) = x$$

è tale per cui $\rho(x) \in D$ (essendo $S \subset D$), inoltre, tenendo conto della (17), vale la (i) della Proposizione 2.

Sia ora $y \in D$. Dobbiamo definire una funzione $\sigma : D \to S$ tale che $\sigma(y) \in S$ e vale la (ii) della Proposizione 2. A questo fine, osserviamo che i due problemi, con un cambio di notazione, possono essere riscritti come segue

$$\begin{aligned}
&\min_{\delta,z} \sum_{i=1}^{N} \delta_i^T d(x^i; z) \\
&e^T \delta_i = 1 \qquad i = 1, \ldots, N \\
&\delta_i \in \{0,1\}^M \qquad i = 1, \ldots, N \\
&z_j \in R^n \qquad j = 1, \ldots, M
\end{aligned} \tag{18}$$

$$\begin{aligned}
&\min_{\delta,z} \sum_{i=1}^{N} \delta_i^T d(x^i; z) \\
&e^T \delta_i = 1 \qquad i = 1, \ldots, N \\
&\delta_i \geq 0 \qquad i = 1, \ldots, N \\
&\delta_i \in R^M \qquad i = 1, \ldots, N \\
&z_j \in R^n \qquad j = 1, \ldots, M
\end{aligned} \tag{19}$$

Fissati i vettori $\bar{z}_j$, per $j = 1, \ldots, M$, il problema (19) rispetto a δ è decomponibile in N problemi separabili, ognuno della forma

$$\begin{aligned}
&\min_{\delta_i} \delta_i^T d(x^i; \bar{z}) \\
&e^T \delta_i = 1 \\
&\delta_i \geq 0
\end{aligned} \tag{20}$$

Il problema (20) ha funzione obiettivo lineare e vincoli di tipo simplesso. Sia $j_{min} \in \{1, \ldots, M\}$ un indice tale che

$$d_{j_{min}}(x^i, \bar{z}_{j_{min}}) \leq d_j(x^i, \bar{z}_j) \quad j = 1, \ldots, M.$$

Dai risultati riportati in Appendice B segue che una soluzione $\delta_i^\star$ di (20) è ottenuta ponendo

$$\delta_{ij}^\star = \begin{cases} 1 & \text{se } j = j_{min} \\ \\ 0 & \text{altrimenti} \end{cases} \tag{21}$$

Il punto $(\bar{z} \quad \delta^\star)^T$ è ammissibile per i problemi (P) e (Q), inoltre, per ogni punto $(\bar{z} \quad \delta)^T$ ammissibile per il problema (Q), possiamo scrivere

$$f(\bar{z}, \delta^\star) = g(\bar{z}, \delta^\star) \leq g(\bar{z}, \delta). \tag{22}$$

Dato un punto $y = (z \quad \delta)^T \in D$, ossia ammissibile per il problema (Q), definiamo

$$\sigma(y) = \begin{pmatrix} z \\ \\ \delta^\star \end{pmatrix},$$

dove $\delta_i^\star$, per $i = 1, \ldots, N$ sono soluzioni di tipo (21 dei problemi (20). Di conseguenza, risulta $\sigma(y) \in S$ e, tenendo conto della (22), $f(\sigma(y)) \leq g(y)$, per cui possiamo concludere che vale la (ii) della Proposizione 2. □

4 Appendice B: Condizioni di ottimalità per problemi con vincoli di simplesso

Sia $x^* \in R^n$ un punto di minimo locale del problema

$$\begin{array}{ll} min & f(x) \\ & \sum_{i=1}^{n} x_i = r \\ & x \geq 0, \end{array} \tag{23}$$

con $r > 0$. Allora risulta

$$x_i^\star > 0 \quad \text{implica } \frac{\partial f(x^\star)}{\partial x_i} \leq \frac{\partial f(x^\star)}{\partial x_j} \text{ per ogni } j \in \{1, \ldots, n\}. \tag{24}$$

Infatti, sia $i \in \{1, \ldots, n\}$ tale che $x_i^\star > 0$ e sia $j \in \{1, \ldots, n\}$ con $j \neq i$. Si verifica facilmente che la direzione d tale che $d_i = -1$, $d_j = 1$, $d_h = 0$, $h \in \{1, \ldots, n\}, h \neq i, j$ è ammissibile in $x^\star$. Essendo $x^\star$ un minimo locale deve necessariamente risultare

$$\nabla f(x^\star)^T d = \frac{\partial f(x^\star)}{\partial x_j} - \frac{\partial f(x^\star)}{\partial x_i} \geq 0.$$

Le condizioni (24) possono essere ricavate dalle condizioni KKT (si lascia per esercizio). Di conseguenza, se la f è convessa le condizioni (24) sono necessarie e sufficienti di minimo globale.

Consideriamo ora un caso particolare di programmazione convessa, ossia il caso di funzione obiettivo lineare:

$$\begin{array}{ll} min & c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ & \sum_{i=1}^{n} x_i = r \\ & x \geq 0, \end{array} \tag{25}$$

Da quanto detto in precedenza segue che una soluzione $x^\star$ si ottiene per via analitica. In particolare, indicato con i l'indice "di costo minimo", ossia tale che

$$c_i \leq c_j \qquad j = 1, \ldots, n,$$

una soluzione $x^\star$ si ottiene ponendo

$$x_i^\star = r \qquad x_j^\star = 0, j \neq i, j = 1, \ldots, n.$$

5 Appendice C: cenni sui metodi incrementali

Si consideri il problema

$$min_{x \in R^n} f(x) = \sum_{i=1}^{m} f_i(x), \tag{26}$$

dove $f_i : R^n \to R$ sono funzioni continuamente differenziabili su R^n. Un caso particolare del problema (26) è un problema di minimi quadrati.

I metodi per la soluzione del problema (26) possono essere distinti in due classi:

- metodi di tipo *batch*, che utilizzano le informazioni riguardanti la funzione "complessiva" $f(x)$ e le sue derivate;
- metodi *incrementali* (anche denominati *online*), che utilizzano le informazioni riguardanti il singolo termine $f_i(x)$ e le sue derivate.

I metodi classici di ottimizzazione (gradiente, Newton, ...) sono ovviamente metodi di tipo batch.

Un metodo incrementale di tipo gradiente può essere descritto come segue. Dato il punto corrente x_k, il nuovo punto x_{k+1} è ottenuto alla fine del seguente ciclo:

$$\begin{array}{l} z_0 = x_k \\ z_i = z_{i-1} - \alpha_k \nabla f_i(z_{i-1}) \quad i = 1, \ldots, m \\ x_{k+1} = z_m, \end{array}$$

dove $\alpha_k > 0$ è il passo di ricerca unidimensionale, e si è omessa la dipendenza di z_i da k per non appesantire la notazione. Due potenziali vantaggi dei metodi di tipo incrementale sono i seguenti:

- possono fornire "buone" soluzioni approssimate nei problemi in cui i dati (da cui dipendono i singoli termini f_i) non sono assegnati *fuori linea*, ma sono generati in tempo reale;

- se il numero di dati è molto elevato (m "grande"), può accadere che i dati stessi siano *statisticamente omogenei*, nel senso che uno stesso vettore rappresenta (approssimativamente) un punto di minimo di molti termini f_i, per cui, una "buona" soluzione del problema potrebbe essere ottenuta dopo un singolo ciclo.

Ossserviamo che

$$x_{k+1} = x_k - \alpha_k \sum_{i=1}^{m} \nabla f_i(z_{i-1}). \tag{27}$$

Un metodo incrementale differisce quindi dal metodo del gradiente per il fatto che utilizza la direzione

$$-\sum_{i=1}^{m} \nabla f_i(z_{i-1})$$

al posto della direzione dell'antigradiente

$$-\sum_{i=1}^{m} \nabla f_i(x_k).$$

Di conseguenza, un metodo incrementale può essere visto come *un metodo del gradiente con errore*. In particolare, si ha

$$x_{k+1} = x_k - \alpha_k \left(\nabla f(x_k) + e_k\right), \tag{28}$$

dove

$$e_k = \sum_{i=1}^{m} \left(\nabla f_i(z_{i-1}) - \nabla f_i(x_k)\right). \tag{29}$$

La scelta del passo α_k è cruciale per assicurare la convergenza globale di un metodo incrementale. Dalla (28) si vede che la direzione di un metodo incrementale differisce dalla direzione del metodo del gradiente di un errore che é proporzionale al passo α_k. Per questa ragione, per assicurare la convergenza globale è essenziale che il passo si riduca iterativamente.

Osserviamo inoltre che, nel caso in cui la sequenza $\{x_k\}$ converge, allora i vettori

$$\alpha_k \nabla f_i(z_{i-1}), \qquad \text{per } i = 1, \ldots, m$$

devono convergere a zero. D'altra parte, poiché non è necessario che i singoli gradienti ∇f_i tendano a zero, segue che è necessario imporre $\alpha_k \to 0$.

Un metodo del gradiente incrementale può essere interpretato come un metodo del gradiente con errore. Sulla base dei risultati di convergenza del metodo del gradiente con errore, è allora possibile garantire, sotto opportune ipotesi, proprietà di convergenza globale di un metodo del gradiente incrementale.

Proposizione 4 (Convergenza del metodo incrementale)
Sia $f(x) = \sum_{i=1}^{m} f_i(x)$, *dove* $f_i : R^n \to R$ *sono funzioni continuamente differenziabili su* R^n. *Sia* $\{x_k\}$ *la sequenza* (27) *generata da un metodo incrementale tipo gradiente. Si assuma l'esistenza di tre costanti positive* L, C, *e* D *tali che per* $i = 1, \ldots, m$ *si ha*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\| \qquad \text{per ogni } x, y \in R^n \tag{30}$$

$$\|\nabla f_i(x)\| \leq C + D\|\nabla f(x)\| \qquad \text{per ogni } x \in R^n. \tag{31}$$

Si assuma inoltre che

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \sum_{k=0}^{\infty} (\alpha_k)^2 < \infty. \tag{32}$$

Allora, o $f(y_k) \to -\infty$, *oppure la sequenza* $\{f(x_k)\}$ *converge ad un valore finito e si ha*

(i) $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$;

(ii) *ogni punto di accumulazione di* $\{x_k\}$ *è un punto stazionario di* f.