Capitolo 1

Clustering

1.1 Introduzione

1.2 K-Means

Il metodo K-Means [**?**] é anche conosciuto col nome di C-means clustering, applicato in diversi contesti, tra i quali la compressione delle immagini e di segnali vocali, ed il riconoscimento di aree tematiche per immagini satellitari.

Consideriamo il problema di identificare gruppi di dati o clusters in uno spazio multidimensionale d. Supponiamo di avere a disposizione il dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ costituito da n osservazioni di un fenomeno fisico d-dimensionale. Il nostro obiettivo é quello di partizionare il nostro dataset in un numero K[1] di gruppi. Ciascuna partizione é rappresentata da un prototipo che in media ha una distanza intra-class[2] piú piccola rispetto a distanze prese tra il prototipo del gruppo ed un'osservazione appartenente ad un altro gruppo. Quindi rappresentiamo con μ_k un vettore d-dimensionale rappresentante il prototipo del k-esimo gruppo (con $k = 1, \ldots, K$). In altre parole, il prototipo rasppresenta il centro del gruppo. Siamo interessati a trovare l'insieme dei prototipi sulla base del dataset X noto tale per cui la somma dei quadrati delle distanze di ciascuna osservazione $\mathbf{x}_i$ con il prototipo piú vicino é minima. Introduciamo ora una notazione per definire la modalitá di assegnare ciascuna osservazione ad un prototipo con la seguente variabile binaria $r_{ik} = 0, 1$ indicante se l'osservazione i-esima appartiene al gruppo k-esimo quindi $r_{ik} = 1$ oppure se $r_{ik} = 0$ se appartiene a qualche altro gruppo diverso da k. Quindi in generale avremo una matrice $\mathbf{R}$ di membership con dimensione $n \times K$ di tipo binaria che evidenzia se il dato i-esimo appartiene alla classe k-esima. Supponiamo per ora di avere i K prototipi $\mu_1, \mu_2, \ldots, \mu_K$ (piú avanti vedremo come calcolarli analiticamente), e pertanto diremo che un'osservazione $\mathbf{x}$ appartiene alla classe k-esima se é soddisfatta la seguente:

$$\| \mathbf{x}^\top - \mu_k \| = \min_j \| \mathbf{x}^\top - \mu_j \| \tag{1.1}$$

che evidenzia il rappresentate per l'osservazione x. A questo punto possiamo valutare l'errore che si commette nell'eleggere il rappresentate di ciascun gruppo, introducendo un funzionale denominato *misura di distorsione* dei dati oppure *errore totale di ricostruzione* con la seguente funzione:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \| \mathbf{x}_i^\top - \mu_k \|^2 \tag{1.2}$$

[1] Supponiamo al momento di conoscere il numero K di gruppi da ricercare.

[2] la distanza tra tutte le osservazioni appartenenti ad uno stesso gruppo con il prototipo rappresentante del gruppo.

Il nostro obiettivo é quindi trovare i valori di $\{r_{ik}\}$ e $\{\mu_k\}$ al fine di minimizzare la funzione obiettivo J. Si fa uso di una procedura iterativa che implica due passi diversi per ciascuna iterazione. Si inizializzano i centri dei cluster $\{\mu_k\}_{k=1} + K$ in maniera casuale. Quindi per ciascuna iterazione, in una prima fase si minimizza J rispetto a r_{ik} e mantenendo fissi i centri μ_k. Nella seconda fase, si minimizza J rispetto μ_k ma mantenendo fissi le funzioni di membership delle classi r_{ik}. Questi due passi sono ripetuti fino a raggiungere la convergenza. Si vedrá in seguito che questi due passi descritti di aggiornamento di r_{ik} e μ_k corrispondono rispettivamente al passo E (expectation) e M (Maximization) dell'algoritmo EM.

La funzione membership r_{ik} se vale 1 ci dice che il vettore $\mathbf{x}_i$ é piú vicino al centro μ_k, ovvero assegnamo ciascun punto del dataset al piú vicino centro del cluster come segue:

$$r_{ik} = \begin{cases} 1 & Se\ k = \arg\min_j \parallel \mathbf{x}_i^\top - \mu_j \parallel \\ 0 & altrimenti. \end{cases} \tag{1.3}$$

Dal momento che una data osservazione x puó appartenere ad un solo gruppo, la matrice $\mathbf{R}$ possiede la seguente proprietá

$$\sum_{k=1}^{K} r_{ik} = 1 \quad \forall i = 1, \ldots, n \tag{1.4}$$

e

$$\sum_{k=1}^{K} \sum_{i=1}^{n} r_{ik} = n. \tag{1.5}$$

Deriviamo ora le formule di aggiornamento per r_{ik} e $mathbf\mu_k$ al fine di minimizzare J. Se consideriamo l'ottimizzazione di $mathbf\mu_k$ rispetto a r_{ik} fissato, si puó notare che la funzione J in (1.2) é una funzione quadratica di $mathbf\mu_k$, che puó essere minimizzata ponendo a zero la derivata prima:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{n} r_{ik} \left(\mathbf{x}_i^\top - \mu_k \right) = 0 \tag{1.6}$$

da cui si ottiene

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}} \tag{1.7}$$

Si noti che il denominatore della (1.7) rappresenta il numero di punti assegnati al k-esimo cluster, ovvero calcola μ_k come media dei punti che ricadono nel cluster. Per questa ragione é chiamato K-means. Finora abbiamo descritto la versione batch dell'algoritmo, in cui tutto il dataset viene usato in una unica soluzione per aggiornare i prototipi, come descritto nell'algoritmo 1. Una versione on-line stocastica dell'algoritmo é stata proposta in letteratura [**?**] applicando la procedura Robbins-Monro al problema di cercare le radici della funzione di regressione data dalle derivate di J rispetto a μ_k. Questo ci permette di formulare una versione sequenziale dell'aggionramento come segue

$$\mu_k^{new} = \mu_k^{old} + \eta_i \left(\mathbf{x}_i - \mu_k^{old} \right) \tag{1.8}$$

con η_i il parametro di apprendimento che viene fatto decrescere monotonicamente sulla base del numero di osservazioni che compongono il dataset. In figura 1.1 é riportato il risultato della quantizzazione o classificazione di pixel a colori. In particolare, in figura 1.1a l'immagine originale e nelle successive il risultato del metodo con diversi valori di prototipi $K = 3, 5, 6, 7, 8$. Ogni colorazione indica un particolare cluster, quindi al pixel originale é stato rimpiazzato il valore (rappresentante la terna RGB del colore) del prototipo piú vicino.

Algorithm 1 Algoritmo K-means

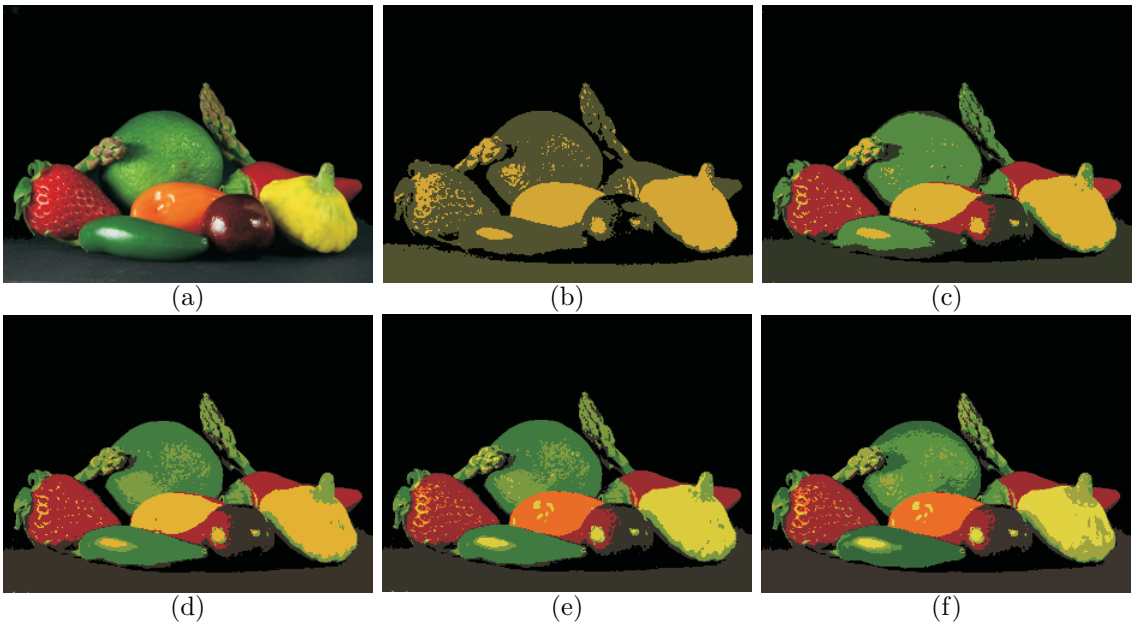
1: Inizializza i centri μ_k per $1 \leq k \leq K$ in maniera casuale
2: **repeat**
3: **for** $\mathbf{x}_i$ per $i = 1, \ldots, n$ **do**
4:

$$r_{ik} = \begin{cases} 1 & Se\ k = \arg\min_j \parallel \mathbf{x}_i^\top - \mu_j \parallel \\ 0 & altrimenti. \end{cases}$$

5: **end for**
6: **for** μ_k per $k = 1, 2, \ldots, K$ **do**
7:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

8: **end for**
9: **until** convergenza dei parametri



(a) (b) (c) (d) (e) (f)

Figura 1.1: Classificazione di pixel RGB con il metodo K-means dell'immagine in (a). In (b) per $K = 3$; (c) per $K = 5$; (d) per $K = 6$; (e) per $K = 7$ e (f) per $K = 8$;

1.2.1 Limitazioni

La soluzione trovata con l'algoritmo, ovvero i protipi delle K classi, possono essere valutate in termini di distanze tra dataset e gli stessi prototipi. Tale informazione di distanza fornisce una misura di distorsione dei dati. Diverse soluzioni sono ottenute con le diverse inizializzazioni che si possono impostare all'algoritmo. Il valore della distorsione puó guidare verso la migliore soluzione trovata sullo stesso dataset di dati, ovvero la minima distorsione. Un'altra limitazione é data dalle variabili di membership o anche dette variabili di responsabilitá z_i^t che assegnano il dato i al cluster t in maniera hard ossia binaria. Nel fuzzy C-means, ma anche nella mistura di Gaussiane trattate

in seguito, queste variabili sono trattate in maniera soft ovvero con valori che variano tra zero e uno.

1.3 Fuzzy C-Means

Questa tecnica proposta da Bezdek [**?**], nota anche col nome di Fuzzy ISODATA, si defferenzia dalla precedente per la funzione di membership. In questo algoritmo ciascun punto x ha una funzione membership r di tipo smooth, ovvero non é binaria ma definisce il grado di appartenenza del dato a ciascun cluster. Questo algoritmo partiziona il dataset X in K gruppi fuzzy, e trova i centri dei cluster in maniera tale da minimizzare la funzione costo di similaritá J. Quindi il partizionamento del dataset é fatto in maniera fuzzy, in modo tale da avere per ciascun dato $x_i \in X$ un valore di appartenenza a ciascun cluster compreso tra 0 e 1. Quindi la matrice di membership $\mathbf{R}$ non é binaria, ma ha valori compresi tra 0 e 1. In ogni caso si impone la condizione (1.4). Quindi la funzione obiettivo diventa

$$J = \sum_{i=1}^{n}\sum_{k=1}^{K} r_{ik}^{m} \parallel \mathbf{x}_i^{\top} - \mu_k \parallel^2 \tag{1.9}$$

con $m \in [1, \infty)$ un esponente che rappresenta un peso. La condizione necessaria per la ricerca del minimo della (1.10) puó essere trovata introducendo i moltiplicatori di Lagrange λ_i che definiscono la seguente funzione costo soggetta agli n vincoli della (1.4):

$$\begin{aligned} \overline{J} &= J + \textstyle\sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K} r_{ik} - 1 \right) \\ &= \textstyle\sum_{k=1}^{K}\sum_{i=1}^{n} r_{ik}^{m} \parallel \mathbf{x}_i^{\top} - \mu_k \parallel^2 + \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K} r_{ik} - 1 \right) \end{aligned} \tag{1.10}$$

Ora differenziando la (1.10) rispetto a μ_k, λ_i e ponendo a zero si perviene alle seguenti:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik}^{m} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}^{m}}, \tag{1.11}$$

e

$$r_{ik} = \frac{1}{\sum_{t=1}^{K} \left(\frac{\|\mathbf{x}_i^{\top} - \mu_k\|}{\|\mathbf{x}_i^{\top} - \mu_t\|} \right)^{2/(m-1)}}. \tag{1.12}$$

Nella versione batch, l'algoritmo é indicato in 2.

Algorithm 2 Algoritmo Fuzzy C-means

1: Inizializza la matrice di membership $\mathbf{R}$ con valori casuali tra 0 e 1 e mantenendo soddisfatta la (1.4)
2: Calcolare i centri dei cluster fuzzy con la (1.11) $\forall\ k = 1, \dots K$.
3: Calcolare la funzione costo in accordo all'equazione (1.10). Valutare il criterio di stop se $J < Soglia$ oppure se la differenza $J_t - J_{t-1} < Soglia$ con t il passo di avanzamento.
4: Calcolare la nuova matrice $\mathbf{R}$ usando la (1.12) e vai al passo 2.

1.4 Misture di Gaussiane (MoG)

La distribuzione Gaussiana possiede alcune limitazioni nel modellare insiemi di dati del mondo reale. Densitá molto complesse possono essere modellate con una combinazione lineare di piú Gaussiane pesate opportunamente. Le applicazioni che fanno uso di queste misture sono molteplici, in particolare, nell'ambito della modellazione del background per una sequenza di streaming video in contesto di videosorveglianza. Il problema che si vuole risolvere é quindi il seguente: dato un

insieme di punti $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ si vuole trovare la distribuzione di probabilitá $p(\mathbf{x})$ che ha generato tale insieme. Nel framework delle misture di Gaussiane, tale distribuzione generatrice del dataset $\mathbf{X}$ é costituito da un insieme composto di K distribuzioni Gaussiane; ciascuna di esse é quindi

$$\mathbf{x}; \theta_k \sim \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k) \tag{1.13}$$

con μ_k e Σ_k rispettivamente la media e la matrice di covarianza della k-esima distribuzione Gaussiana, definita come segue:

$$\mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k) = (2\pi)^{-d/2}\Sigma^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right\} \tag{1.14}$$

ovvero la probabilitá di generare l'osservazione $\mathbf{x}$ mediante il modello k-esimo é fornita dalla distribuzione Gaussiana avente il vettore dei parametri $\theta_k = (\mu_k, \Sigma_k))$ (media e matrice di covarianza rispettivamente). Introduciamo in questo modello una variabile ulteriore $\mathbf{z}_i$ associata all'osservazione $\mathbf{x}_i$. La variabile $\mathbf{z}_i$ é una variabile che indica quale dei K modelli ha generato il dato $\mathbf{x}_i$ in accordo alla probabilitá a priori $\pi_k = p(z_i = k)$. Risulta semplice pensare alla variabile $\mathbf{z}_i$ come un vettore binario K-dimensionale avente un 1 nell'elemento corrispondente alla componente Gaussiana che ha generato il corrispondente dato $\mathbf{x}_i$

$$\mathbf{z}_i = [\underbrace{0\ 0\ \ldots\ 0\ 1\ 0\ \ldots\ 0\ 0}_{K\ elementi}]^\top \tag{1.15}$$

Dal momento che non conosciamo i corrispondenti $\mathbf{z}_i$ per ciascun $\mathbf{x}_i$[3] queste variabili sono denominate variabili *nascoste* (o *hidden*). Il nostro problema ora é ridotto alla ricerca dei parametri μ_k e $\mathbf{\Sigma}_k$ per ciascuno dei K modelli e delle rispettive probabilitá a priori π_k che se incorporato nel modello generativo, ha una probabilitá alta di generare la distribuzione di dati osservata. La densitá della mistura é data da

$$p(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) \tag{1.16}$$

Ciascuna densitá Gaussiana $P_{ik} = p(x_i|z_i = k, \mu_k, \mathbf{\Sigma}_k) = \mathcal{N}(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k)$ rappresenta una *componente* della mistura avente media μ_k e covarianza $\mathbf{\Sigma}_k$. I parametri π_k sono denominati *coefficienti della mistura* e servono a pesare la relativa Gaussiana nel modellare la generica variabile aleatoria $\mathbf{x}$. Se si integrano entrambi i lati della (1.16) rispetto ad x, si puó notare che sia $p(\mathbf{x})$ che le componenti individuali delle Gaussiane sono normalizzati, ottenendo

$$\sum_{k=1}^{K} \pi_k = 1. \tag{1.17}$$

Si noti anche che sia $p(x) \geq 0$ che $\mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) \geq 0$ risultando quindi $\pi_k \geq 0$ per ogni k. Combinando queste ultime condizioni con la (1.17) otteniamo

$$0 \leq \pi_k \leq 1 \tag{1.18}$$

e quindi i coefficienti della mistura sono delle probabilitá.

Siamo interessati a massimizzare la verosimiglianza $\mathcal{L}(\theta) = p(\mathbf{X}; \theta)$ che genera i dati osservati con i parametri del modello $\theta = \{\mu_k, \mathbf{\Sigma}_k, \pi_k\}_{k=1}^K$. Questo approccio é denominato stima di massima verosimiglianza (ML[4]) dal momento che trova i parametri massimizzando la funzione di verosimiglianza che genera i dati. Un approccio alternativo é dato dall'algoritmo EM, dove quest'ultimo

[3] Se li conoscessimo, raggrupperemmo tutti gli $\mathbf{x}_i$ in base ai rispettivi $\mathbf{z}_i$ e modelleremmo ciascun raggruppamento con una singola Gaussiana.

[4] Maximum Likelihood estimate.

calcola θ massimizzando la probabilitá a posteriori (MAP[5]) e che presenta una trattazione matematica piú trattabile. Quindi riassumento il nostro vettore puó essere stimato nei due seguenti modi:

$$\begin{array}{rcl} \theta_{ML} & = & \arg\max_\theta P(\mathbf{X}|\theta) \\ \theta_{MAP} & = & \arg\max_\theta P(\theta|\mathbf{X}) \end{array} \tag{1.19}$$

1.4.1 Stima dei parametri della mistura con la Massima Verosimiglianza (ML)

Il nostro obbiettivo é quindi trovare i parametri delle K distribuzioni Gaussiane ed i coefficienti π_k sulla base del dataset $\mathbf{X}$ di cui disponiamo. Calcolando la densitá sull'intero dataset di misure statisticamente indipendenti abbiamo:

$$p(X|\theta) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \; p(x_i|z_i = k, \mu_k, \mathbf{\Sigma}_k) \tag{1.20}$$

Per trovare il vettore dei parametri θ un modo é rappresentato dalla stima di massima verosimiglianza. Calcolando il logaritmo della (1.20) otteniamo

$$\mathcal{L} = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \; p(x_i|z_i = k, \mu_k, \mathbf{\Sigma}_k) \tag{1.21}$$

che differenzieremo rispetto a $\theta = \{\mu, \Sigma\}$ e annulleremo per trovare il massimo, ovvero

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_{i=1}^{n} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j P_{ij}} \frac{\partial P_{ik}}{\partial \theta_k} = \sum_{i=1}^{n} \underbrace{\frac{\pi_k P_{ik}}{\sum_{j=1}^{K} \pi_j P_{ij}}}_{r_{ik}} \frac{\partial \log P_{ik}}{\partial \theta_k} = \sum_{i=1}^{n} r_{ik} \frac{\partial \log P_{ik}}{\partial \theta_k} \tag{1.22}$$

in cui abbiamo usato l'identitá $\partial p/\partial\theta = p \times \partial \log p/\partial\theta$ e definito r_{ik} le *responsabilitá*, ossia la variabile rappresentante quanto probabile il punto i-esimo sia modellato o spiegato dalla Gaussiana k-esima, ovvero

$$r_{ik} = \frac{p(\mathbf{x}_i, z_i = k|\mu_k, \mathbf{\Sigma}_k)}{p(\mathbf{x}_i|\mu_k, \mathbf{\Sigma}_k)} = p(z_i = k|\mathbf{x}_i, \mu_k, \mathbf{\Sigma}_k) \tag{1.23}$$

che sono probabilitá a posteriori di appartenenza alla classe con $\sum_{k=1}^{K} r_{ik} = 1$. Ora ci rimane da calcolare la derivata rispetto a π_k. Prendiamo la funzione obbiettivo $\mathcal{L}$ ed aggiungiamo un moltiplicatore di Lagrange λ che rafforzano il fatto che le probabilitá a priori necessitano di adattarsi ad 1

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda(1 - \sum_{k=1}^{K} \pi_k) \tag{1.24}$$

quindi deriviamo rispetto a π_k e λ e poniamo a zero come prima

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \pi_k} = \sum_{i=1}^{n} \frac{P_{ik}}{\sum_{j=1}^{n} \pi_j P_{ij}} - \lambda = 0 \Leftrightarrow \sum_{i=1}^{n} r_{ik} - \lambda \pi_k = 0 \tag{1.25}$$

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \lambda} = 1 - \sum_{k=1}^{K} \pi_k = 0 \tag{1.26}$$

[5]Maximum A Posteriori estimate.

e considerando che $\sum_{k=1}\sum_{i=1}^{n} r_{ik} - \lambda\pi_k = n - \lambda = 0$ otteniamo $\lambda = n$, quindi la probabilitá a priori per il k-esimo cluster é data da

$$\pi_k = \frac{1}{n}\sum_{i=1}^{n} r_{ik} \tag{1.27}$$

Troviamo ora media e covarianza dalla funzione obbiettivo $\mathcal{L}$ come segue

$$\frac{\partial \mathcal{L}}{\partial \mu_k} = \sum_{i=1}^{n} r_{ik} \tag{1.28}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Sigma}_k^{-1}} = \sum_{i=1}^{n} r_{ik}[\Sigma_k - (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top] \tag{1.29}$$

Ponendo quindi (1.28) e (1.29) a zero ricaviamo

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}} \qquad \Sigma_k = \frac{\sum_{i=1}^{n} r_{ik}(\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top}{\sum_{i=1}^{n} r_{ik}} \tag{1.30}$$

Si noti che i denominatori $\sum_{i=1}^{n} r_{ik} = n\pi_k$ rappresentano il numero totale di punti assegnati al k-esimo cluster. Inoltre, la media in 1.30 é simile a quella ottenuta per il metodo k-means eccetto per le responsabilitá r_{ik} che in questo caso sono soft (ovvero $0 \leq r_{ik} \leq 1$).

1.4.2 Stima dei parametri della mistura con Expectation Maximization (EM)

Una maniera elegante per trovare stime di massima verosimiglianza per modelli di variabili latenti é caratterizzata dall'algoritmo expectation-maximization o EM [**?**].

Invece di trovare la stima di massima verosimiglianza (ML) dei dati osservati $p(\mathbf{X};\theta)$ cercheremo di massimizzare la verosimiglianza della distribuzione congiunta di $\mathbf{X}$ e di $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{n}$, $p(\mathbf{X},\mathbf{Z};\theta)$. A tal proposito preferiamo massimizzare il logaritmo della verosimiglianza, ossia

$$l_c(\theta) = \log p(\mathbf{X},\mathbf{Z};\theta),$$

quantitá conosciuta col nome di *complete log-likelihood*. Dal momento che noi non possiamo osservare i valori delle variabili aleatorie $\mathbf{z}_i$, dobbiamo lavorare con i valori attesi della quantitá $l_c(\theta)$ rispetto a qualche distribuzione $Q(\mathbf{Z})$. Il logaritmo della funzione di verosimiglianza completa é definito come segue:

$$\begin{aligned} l_c(\theta) &= \log p(\mathbf{X},\mathbf{Z};\theta) \\ &= \log \textstyle\prod_{i=1}^{n} p(\mathbf{x}_i,\mathbf{z}_i;\theta) \\ &= \log \textstyle\prod_{i=1}^{n}\prod_{k=1}^{K} [p(\mathbf{x}_i|z_{ik}=1;\theta)p(z_{ik}=1)]^{z_{ik}} \\ &= \textstyle\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik}\log p(\mathbf{x}_i|z_{ik}=1;\theta) + z_{ik}\log\pi_k. \end{aligned} \tag{1.31}$$

Poiché abbiamo assunto che ciascuno dei modelli é una Gaussiana, la quantitá $p(\mathbf{x}_i|k,\theta)$ rappresenta la probabilitá condizionata nel generare $\mathbf{x}_i$ dato il modello k-esimo:

$$\log p(\mathbf{x}_i|z_{ik}=1;\theta) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1}(\mathbf{x}_i - \mu_k)\right\} \tag{1.32}$$

Considerando il valore atteso rispetto a $Q(\mathbf{Z})$

$$\langle l_c(\theta)\rangle_{Q(\mathbf{Z})} = \sum_{i=1}^{n}\sum_{k=1}^{K}\langle z_{ik}\rangle \log p(\mathbf{x}_i|z_{ik}=1;\theta) + \langle z_{ik}\rangle \log\pi_k. \tag{1.33}$$

Il passo M

Il passo M considera il valore atteso della funzione $l_c(\theta)$ definito in (1.33) e la massimizza rispetto ai parametri stimati nell'altro passo (quello E di aspettazione) che sono quindi π_k, μ_k e Σ_k. Differenziando l'equazione (1.33) rispetto a μ_k e ponendo a zero otteniamo:

$$\frac{\partial \langle l_c(\theta) \rangle_{Q(\mathbf{Z})}}{\partial \mu_k} = \sum_{i=1}^{n} \langle z_{ik} \rangle \frac{\partial}{\partial \mu_k} \log p(\mathbf{x}_i | z_{ik} = 1; \theta) = \mathbf{0} \tag{1.34}$$

Possiamo ora calcolare $\frac{\partial}{\partial \mu_k} \log p(\mathbf{x}_i | z_{ik} = 1; \theta)$ usando la (1.32) come segue:

$$\begin{aligned} \tfrac{\partial}{\partial \mu_k} \log p(\mathbf{x}_i | z_{ik} = 1; \theta) &= \tfrac{\partial}{\partial \mu_k} \log \left\{ \tfrac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\tfrac{1}{2} (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \right\} \right\} \\ &= -\tfrac{1}{2} \tfrac{\partial}{\partial \mu_k} (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \\ &= (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} \end{aligned} \tag{1.35}$$

in cui l'ultima eguaglianza deriva dalla relazione $\frac{\partial}{\partial x} \mathbf{x}^\top \mathbf{A} \mathbf{x} = \mathbf{x}^\top (\mathbf{A} + \mathbf{A}^\top)$. Sostituendo il risultato della (1.35) nella (1.34) otteniamo:

$$\sum_{i=1}^{n} \langle z_{ik} \rangle (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} = 0 \tag{1.36}$$

la quale ci fornisce l'equazione di aggiornamento

$$\mu_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle \mathbf{x}_i}{\sum_{i=1}^{n} \langle z_{ik} \rangle} \tag{1.37}$$

Calcoliamo ora la stima per la matrice di covarianza differenziando l'equazione (1.33) rispetto a Σ_k^{-1} abbiamo:

$$\frac{\partial \langle l_c(\theta) \rangle_{Q(\mathbf{Z})}}{\partial \Sigma_k^{-1}} = \sum_{i=1}^{n} \langle z_{ik} \rangle \frac{\partial}{\partial \Sigma_k^{-1}} \log p(\mathbf{x}_i | z_{ik} = 1; \theta) = \mathbf{0} \tag{1.38}$$

Possiamo calcolare $\frac{\partial}{\partial \Sigma_k^{-1}} \log p(\mathbf{x}_i | z_{ik} = 1; \theta)$ utilizzando la (1.32) come segue:

$$\begin{aligned} \tfrac{\partial}{\partial \Sigma_k^{-1}} \log p(\mathbf{x}_i | z_{ik} = 1; \theta) &= \tfrac{\partial}{\partial \Sigma_k^{-1}} \log \left\{ \tfrac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\tfrac{1}{2} (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \right\} \right\} \\ &= \tfrac{\partial}{\partial \Sigma_k^{-1}} \left\{ \tfrac{1}{2} \log |\Sigma_k^{-1}| - \tfrac{1}{2} (\mathbf{x}_i - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \right\} \\ &= \tfrac{1}{2} \Sigma_k - \tfrac{1}{2} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top \end{aligned} \tag{1.39}$$

dove l'ultima eguaglianza é ottenuta dalla relazione $\frac{\partial}{\partial \mathbf{X}} \log |\mathbf{X}| = (\mathbf{X}^{-1})^\top$ e $\frac{\partial}{\partial \mathbf{A}} \mathbf{x}^\top \mathbf{A} \mathbf{x} = \mathbf{x}\mathbf{x}^\top$). Sostituendo questo risultato nella (1.38) otteniamo:

$$\sum_{i=1}^{n} \langle z_{ik} \rangle \left(\frac{1}{2} \Sigma_k - \frac{1}{2} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top \right) = \mathbf{0} \tag{1.40}$$

che ci fornisce l'equazione di aggiornamento della matrice di covarianza per la k-esima componente della mistura:

$$\Sigma_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top}{\sum_{i=1}^{n} \langle z_{ik} \rangle} \tag{1.41}$$

Ora ci serve trovare l'equazione di aggiornamento della probabilitá a priori π_k per la k-esima componente della mistura. Questo significa massimizzare il valore atteso della funzione logaritmo

della verosimiglianza $\langle l_c \rangle$ (1.33) soggetta al vincolo che $\sum_k \pi_k = 1$. Per far ció, introduciamo i moltiplicatori di Lagrange λ aumentando la (1.33) come segue:

$$\mathcal{L}(\theta) = \langle l_c(\theta) \rangle_{Q(\mathbf{Z})} - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right). \tag{1.42}$$

Differenziando questa espressione rispetto a π_k otteniamo

$$\frac{\partial}{\partial \pi_k} \langle l_c(\theta) \rangle_{Q(\mathbf{Z})} - \lambda = 0 \qquad 1 \leq k \leq K. \tag{1.43}$$

Utilizzando la (1.33) abbiamo:

$$\left.\begin{array}{rr} & \frac{1}{\pi_k} \sum_{i=1}^{n} \langle z_{ik} \rangle - \lambda = 0 \\ o \;\; equivalentemente & \sum_{i=1}^{n} \langle z_{ik} \rangle - \lambda \pi_k = 0 \end{array}\right\} \quad 1 \leq k \leq K \tag{1.44}$$

Sommando ora l'equazione (1.44) su tutti i K modelli si ha:

$$\sum_{k=1}^{K} \sum_{i=1}^{n} \langle z_{ik} \rangle - \lambda \sum_{k=1}^{K} \pi_k = 0 \tag{1.45}$$

e poiché $\sum_{k=1}^{K} \pi_k = 1$ abbiamo:

$$\lambda = \sum_{k=1}^{K} \sum_{i=1}^{n} \langle z_{ik} \rangle = n \tag{1.46}$$

Sostituendo questo risultato nell'equazione (1.44) otteniamo la seguente formula di aggiornamento:

$$\pi_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle}{n} \tag{1.47}$$

che conserva il vincolo $\sum_{k=1}^{K} \pi_k = 1$.

Il passo E

Ora che abbiamo derivato le formule di aggiornamento dei parametri che massimizzano il valore atteso della funzione completa di verosimiglianza $\log p(\mathbf{X}, \mathbf{Z}; \theta)$ dobbiamo anche assicurarci che stiamo massimizzando la versione incompleta $\log p(\mathbf{X}; \theta)$ (che di fatto é la quantitá che ci interessa massimizzare realmente). Come accennato in precedenza, siamo sicuri di massimizzare la versione incompleta del logaritmo della verosimiglianza, solo quando si considera il valore atteso della distribuzione a posteriori della $\mathbf{Z}$, ovvero $p(\mathbf{Z}|\mathbf{X}; \theta)$. Quindi, ciascuno dei valori attesi $\langle z_{ik} \rangle$ che appaiono nell'equazione di aggiornamento derivata nel precedente paragrafo, dovrebbe essere calcolato come segue:

$$\begin{aligned} \langle z_{ik} \rangle_{p(\mathbf{Z}|\mathbf{X};\theta)} &= 1 \cdot p(z_{ik} = 1|\mathbf{x}_i; \theta) + 0 \cdot p(z_{ik} = 0|\mathbf{x}_i; \theta) \\ &= p(z_{ik} = 1|\mathbf{x}_i; \theta) \\ &= \frac{p(\mathbf{x}_i|z_{ik}=1;\theta) p(z_{ik}=1)}{\sum_{j=1}^{K} p(\mathbf{x}_i|z_{ij}=1;\theta) p(z_{ij}=1)} \\ &= \frac{p(\mathbf{x}_i|z_{ik}=1;\theta) \pi_k}{\sum_{j=1}^{K} p(\mathbf{x}_i|z_{ij}=1;\theta) \pi_j} \end{aligned} \tag{1.48}$$

Algorithm 3 Algoritmo EM per misture di Gaussiane

Inizializza $\langle z_{ik} \rangle$, π_k, μ_k e Σ_k per $1 \leq k \leq K$
repeat
  for $i = 1, 2, \ldots, n$ **do**
    for $k = 1, 2, \ldots, K$ **do**

$$p(\mathbf{x}_i | z_{ik} = 1; \theta) = (2\pi)^{-d/2} \Sigma^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) \right\}$$

$$\langle z_{ik} \rangle = \frac{p(\mathbf{x}_i | z_{ik} = 1; \theta) \pi_k}{\sum_{j=1}^{K} p(\mathbf{x}_i | z_{ij} = 1; \theta) \pi_j}$$

    end for
  end for
  for $k = 1, 2, \ldots, K$ **do**

$$\Sigma_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top}{\sum_{i=1}^{n} \langle z_{ik} \rangle}$$

$$\mu_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle \mathbf{x}_i}{\sum_{i=1}^{n} \langle z_{ik} \rangle}$$

$$\pi_k = \frac{\sum_{i=1}^{n} \langle z_{ik} \rangle}{n}$$

  end for
until convergenza dei parametri

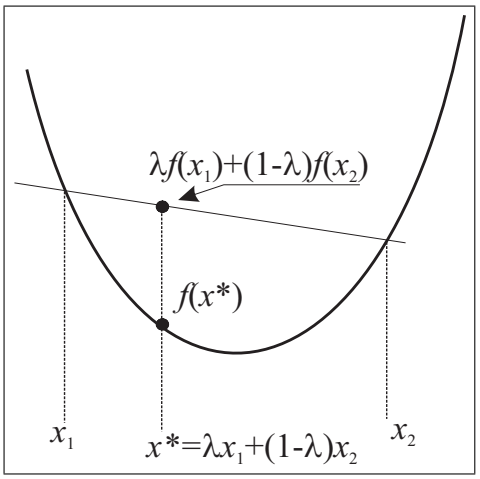


Figura 1.2: Rappresentazione di una funzione convessa per la diseguaglianza di Jensen

1.4.3 Teoria EM

Poiché risulta difficile massimizzare analiticamente la quantitá $\log p(\mathbf{X}; \theta)$ si sceglie pertanto di massimizzare la versione completa del logaritmo della verosimiglianza $\log p(\mathbf{X}, \mathbf{Z}; \theta)_{Q(\mathbf{Z})}$ ricorrendo ad un teorema noto col nome di *diseguaglianza di Jensen*. Il nostro obbiettivo é pertanto massimiz-

zare $\log p(\mathbf{X}, \mathbf{Z}; \theta)_{Q(\mathbf{Z})}$ con la speranza di massimizzare anche la sua versione incompleta $\log p(\mathbf{X}; \theta)$ (che di fatto rappresenta la quantitá che ci interessa massimizzare).

Prima di addentrarci in questa giustificazione, introduciamo la seguente diseguaglianza

Diseguaglianza di Jensen Se $f(x)$ é una funzione convessa (figura 1.2), definita in un intervallo (a, b) allora

$$\forall x_1, x_2 \in (a, b)\ \forall \lambda \in [0, 1]\ :\ \lambda f(x_1) + (1 - \lambda) f(x_2) \geq f[\lambda x_1 + (1 - \lambda) x_2]. \tag{1.49}$$

Se alternativamente $f(x)$ é una funzione concava, allora

$$\lambda f(x_1) + (1 - \lambda) f(x_2) \leq f[\lambda x_1 + (1 - \lambda) x_2] \tag{1.50}$$

Ovvero, se voglio trovare il valore della funzione tra i due punti x_1 ed x_2, diciamo $x^* = \lambda x_1 + (1 - \lambda) x_2$, allora il valore di $f(x^*)$ si troverá al di sotto della congiungente $f(x_1)$ ed $f(x_2)$ (nel caso sia convessa, viceversa se concava).

Noi siamo interessati a valutare il logaritmo che di fatto é una funzione concava e per cui prenderemo in considerazione l'ultima diseguaglianza (1.50).

Riscriviamo $\log p(\mathbf{X}; \theta)$ come segue

$$\log p(\mathbf{X}; \theta) = \log \int p(\mathbf{X}, \mathbf{Z}; \theta) d\mathbf{Z} \tag{1.51}$$

Ora moltiplichiamo e dividiamo per una distribuzione arbitraria $Q(\mathbf{Z})$ al fine di trovare un estremo inferiore del $\log p(\mathbf{X}; \theta)$, ed usiamo il risultato della diseguaglianza di Jensen per proseguire la (1.51)

$$\begin{aligned}
\log \textstyle\int p(\mathbf{X}, \mathbf{Z}; \theta) d\mathbf{Z} &= \log \textstyle\int Q(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z}; \theta)}{Q(\mathbf{Z})} d\mathbf{Z} \\
&\geq \textstyle\int Q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z}; \theta)}{Q(\mathbf{Z})} d\mathbf{Z} \quad \textit{Jensen} \\
&= \underbrace{\textstyle\int Q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z}; \theta) d\mathbf{Z}}_{\textit{valore atteso log-verosimiglianza}} - \underbrace{\textstyle\int Q(\mathbf{Z}) \log Q(\mathbf{Z}) d\mathbf{Z}}_{\textit{Entropia di } Q(\mathbf{Z})} \\
&= \langle \log p(\mathbf{X}, \mathbf{Z}; \theta) \rangle_{Q(\mathbf{Z})} + \mathcal{H}[Q(\mathbf{Z})] \\
&= \mathcal{F}(Q, \theta)
\end{aligned} \tag{1.52}$$

Quindi perveniamo al seguente estremo inferiore della funzione $\log p(\mathbf{X}; \theta)$

$$\log p(\mathbf{X}; \theta) \geq \mathcal{F}(Q, \theta) \tag{1.53}$$

Dal momento che $Q(\mathbf{Z})$ é una distribuzione arbitraria, essa é indipendente da θ, e quindi, per massimizzare il funzionale $\mathcal{F}(Q, \theta)$ é sufficiente massimizzare $\langle \log p(\mathbf{X}, \mathbf{Z}; \theta) \rangle_{Q(\mathbf{Z})}$ (da qui lo step M).

Anche se abbiamo trovato un estremo inferiore $\mathcal{F}(Q, \theta)$ per la funzione $\log p(\mathbf{X}; \theta)$, questo non implica che ad ogni passo il miglioramento nella ricerca del massimo di $\mathcal{F}$ si ripercuote nel miglioramento del massimo di $\log p(\mathbf{X}; \theta)$. Se invece, impostiamo $Q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}; \theta)$ nella (1.52)

possiamo osservare che l'estremo inferiore diviene un minimo, ossia una eguaglianza come segue

$$\begin{aligned} \int Q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z};\theta)}{Q(\mathbf{Z})} d\mathbf{Z} &= \int p(\mathbf{Z}|\mathbf{X};\theta) \log \frac{p(\mathbf{X},\mathbf{Z};\theta)}{p(\mathbf{Z}|\mathbf{X};\theta)} d\mathbf{Z} \\ &= \int p(\mathbf{Z}|\mathbf{X};\theta) \log \frac{p(\mathbf{Z}|\mathbf{X};\theta)p(\mathbf{X};\theta)}{p(\mathbf{Z}|\mathbf{X};\theta)} d\mathbf{Z} \\ &= \int p(\mathbf{Z}|\mathbf{X};\theta) \log p(\mathbf{X};\theta) d\mathbf{Z} \\ &= \log p(\mathbf{X};\theta) \int p(\mathbf{Z}|\mathbf{X};\theta) d\mathbf{Z} \\ &= \log p(\mathbf{X};\theta) \end{aligned} \tag{1.54}$$

Questo significa che quando calcoliamo il valore atteso della versione completa della funzione logaritmo della verosimiglianza $\langle \log p(\mathbf{X}, \mathbf{Z}; \theta) \rangle_{Q(\mathbf{Z})}$, esso dovrebbe essere preso rispetto alla probabilitá vera a posteriori $(\mathbf{Z}|\mathbf{X};\theta)$ delle variabili hidden (da qui il passo E).

Assumiamo un modello avente: variabili osservabili (o visibili) $\mathbf{x}$; variabili non osservabili (hidden o nascoste) $\mathbf{y}$; ed il relativo vettore dei parametri θ. Il nostro obbiettivo é massimizzare il logaritmo della verosimiglianza rispetto alla variabile θ contenente i parametri:

$$\mathcal{L}(\theta) = \log p(\mathbf{x}|\theta) = log \int p(\mathbf{x}, \mathbf{y}|\theta) d\mathbf{y} \tag{1.55}$$

Ora moltiplichiamo e dividiamo per una stessa distribuzione arbitraria $q(\mathbf{y})$ definita sulle variabili latenti $\mathbf{y}$. Ora possiamo sfruttare la diseguaglianza di Jensen, poiché abbiamo una combinazione convessa pesata da $q(\mathbf{y})$ di una qualche combinazione di funzioni. In pratica consideriamo come $f(y)$ una funzione di variabili latenti $\mathbf{y}$ come indicato nella (1.56). Qualsiasi distribuzione $q(\mathbf{y})$ sulle variabili nascoste puó essere usata per ottenere un limite inferiore del logaritmo della funzione di verosimiglianza:

$$\mathcal{L}(\theta) = \log \int q(\mathbf{y}) \frac{p(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})} d\mathbf{y} \geq \int q(\mathbf{y}) \log \frac{p(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})} d\mathbf{y} = \mathcal{F}(q, \theta) \tag{1.56}$$

Questo limite inferiore é chiamato diseguaglianza di Jensen e deriva dal fatto che la funzione logaritmo é concava[6]. Nell'algoritmo EM ottimizziamo alternativamente $\mathcal{F}(q, \theta)$ rispetto a $q(\mathbf{y})$ e θ. Si puó provare che questo modo di operare non decrescerá mai $\mathcal{L}(\theta)$.

Riassumendo, l'algoritmo EM si alterna tra i due seguenti passi:

E step ottimizza $\mathcal{F}(Q, \theta)$ rispetto alla distribuzione delle variabili hidden mantenendo i parametri fissi:

$$Q^{(k)}(\mathbf{z}) = \arg\max_{Q(\mathbf{z})} \mathcal{F}(Q(\mathbf{z}), \theta^{(k-1)}). \tag{1.57}$$

M step massimizza $\mathcal{F}(Q, \theta)$ rispetto ai parametri mantenendo fissa la distribuzione delle variabili hidden:

$$\theta^{(k)} = \arg\max_{\theta} \mathcal{F}(Q(\mathbf{z}), \theta^{(k-1)}) = \arg\max_{\theta} \int Q^{(k)}(\mathbf{z}) \log p(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z} \tag{1.58}$$

dove la seconda eguaglianza deriva dal fatto che l'entropia di $q(\mathbf{z})$ non dipende direttamente da θ.

L'intuizione che é alla base dell'algoritmo EM, lo si puó schematizzare come segue

- E-step: trova i valori delle variabili hidden in accordo alle loro probabilitá a posteriori;
- M-step: apprende il modello come se le variabili hidden non fossero hidden (nascoste).

[6]Il logaritmo della media é maggiore delle medie dei logaritmi.

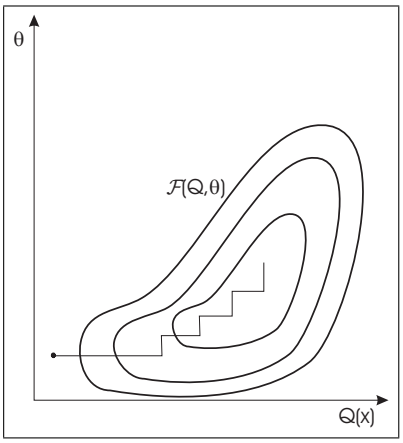


Figura 1.3:

L'algoritmo EM é molto utile in svariati contesti, poiché in molti modelli, se le variabili hidden non lo sono piú, l'apprendimento diviene molto semplice (caso delle misture di Gaussiane). Inoltre, l'algoritmo scompone il problema complesso di apprendimento in una sequenza di problemi di apprendimento piú semplici.

1.5 Mean-Shift

Le tecniche precedentemente viste usano modelli paramterici, ovvero, essi assumono che la densità da ricercare è rappresentata dalla sovrapposizione di distribuzioni elementari (Gaussiane) le cui posizioni (centri) e forme (covarianze) vengono stimate iterativamente. Il mean shift invece, livella la distribuzione dei dati e trova i suoi picchi e le corrispondenti regioni. Mean shift quindi modella la distribuzione usando una forma non-parametrica, ovvero, non fa alcuna assunzione circa la forma della distribuzione in analisi.

Siano $\mathbf{x}_i$ per $i = 1, \ldots, n$ i punti del dataset di uno spazio d-dimensionale di $\Re^d$. Siamo quindi interessati di individuare i prototipi dei possibili cluster presenti dal momento che la nostra distribuzione (come avviene in molti casi reali), sia multivariata. Prima di ciò, è necessario stimare la densità, per poi individuare le mode presenti. La tecnica più nota di stima di densità non-parametrica è caratterizzata dalle finestre di Parzen [**?**]. La probabilità che un vettore $\mathbf{x}$ cadrà in una regione $\mathcal{R}$ è data da

$$P = \int_{\mathcal{R}} f(\mathbf{x}')d\mathbf{x}'. \tag{1.59}$$

Se invece abbiamo n campioni come indicato sopra, e vogliamo calcolare la probabilità che k degli n cadrà nella regione $\mathcal{R}$, sarà data dalla legge binomiale[7], in cui il valore atteso è dato da

$$E[k] = nP, \tag{1.60}$$

in cui la distribuzione binomiale per k risulta essere molto piccata attorno al valore medio con la conseguenza che il rapporto k/n è una stima accurata di P per $n \to \infty$. Se ora assumiamo che $f(\mathbf{x})$ è continua e che la regione $\mathcal{R}$ è talmente piccola al punto da non avere variazioni di f, possiamo scrivere

$$\int_{\mathcal{R}} f(\mathbf{x}')d\mathbf{x}' \simeq f(\mathbf{x})V \tag{1.61}$$

[7]Sotto la condizione che $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ siano variabili aleatorie indipendenti e identicamente distribuite.

con V il volume racchiuso da $\mathcal{R}$. Combinando le (1.59), (1.60) e (1.61) arriviamo alla seguente stima

$$f(\mathbf{x}) \simeq \frac{k/n}{V}. \tag{1.62}$$

Ora però ci sono alcuni problemi pratici e teorici che conducono proprio all'uso delle finestre di Parzen. Ovvero, se fissiamo V e prendiamo molti più campioni di training, il rapporto k/n convergerà in probabilità come desiderato, ma in tal caso avremmo ottenuto solo una stima mediata della densità $f(\mathbf{x})$,

$$\frac{P}{V} = \frac{\int_{\mathcal{R}} f(\mathbf{x}')d\mathbf{x}'}{\int_{\mathcal{R}} d\mathbf{x}'}. \tag{1.63}$$

Se vogliamo ottenere $f(\mathbf{x})$ piuttosto che una sua versione media, dobbiamo imporrre che $V \to 0$. Con questo vincolo, può accadere che se fissiamo il numero di campioni n e facciamo tendere V a zero, la regione diverrà talmente piccola da non avere alcun punto in essa, producendo una stima $f(\mathbf{x}) \simeq 0$ che sarà inutile. Allo stesso modo, può accadere che uno o più campioni concida proprio con $\mathbf{x}$) facendo divergere a infinito la stima. Per ovviare a questi inconvenienti, formiamo una sequenza di partizionamenti del nostro feature space in tante regioni $\mathcal{R}_1, \mathcal{R}_2, \ldots$ contenenti uno o più punti $f(\mathbf{x})$. Ovvero, il pedice individua il numero di campioni che cadono nella rispettiva regione, quindi $\mathcal{R}_1$ avrà un solo dato, $\mathcal{R}_2$ ne avrà due e così via. Indichiamo con V_n il volume contenuto in $\mathcal{R}_n$, e k_n il numero di campioni che ricadono in $\mathcal{R}_n$, e $f_n(\mathbf{x})$ la stima n-esima di $f(\mathbf{x})$

$$f_n(\mathbf{x}) = \frac{k_n/n}{V_n}. \tag{1.64}$$

Affinchè $f_n(\mathbf{x}) \to f(\mathbf{x})$ devono essere verificate tre condizioni

$$\begin{array}{l} \lim_{n\to\infty} V_n = 0 \\ \lim_{n\to\infty} k_n = \infty \\ \lim_{n\to\infty} k_n/n = 0. \end{array}$$

La prima condizione ci assicura che P/V converge a $f(\mathbf{x})$, la seconda condizione ci assicura che il rapporto frequenziale della (1.64) convergerà a P (per $f(\mathbf{x}) \neq 0$), e la terza condizione è necessaria per la convergenza di $f_n(\mathbf{x})$. Le finestre di Parzen usano un modo di ottenere sequenze di regioni che soddisfano queste condizioni, ossia, mediante la riduzione del volume come funzione del numero di campioni $V_n = 1/\sqrt{n}$.

Considerando che nello spazio delle feature $\Re^d$ il volume V_n che racchiude la regione $\mathcal{R}_n$ è caratterizzato da un ipercubo di lato h_n, abbiamo

$$V_n = h_n^d. \tag{1.65}$$

Il numero k_n di campioni che cadono nell'ipercubo è dato dalla seguente funzione:

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq 1 \quad j = 1, \ldots, d \\ 0 & altrimenti \end{cases} \tag{1.66}$$

che rappresenta una finestra definita sull'ipercubo unitario e centrato nell'origine. Se un campione $\mathbf{x}_i$ cade nell'ipercubo, allora $\varphi((\mathbf{x}-\mathbf{x}_i)/h_n) = 1$, altrimenti sarà zero. Quindi il numero di campioni che cade nell'ipercubo saranno

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) \tag{1.67}$$

che sostituendo nella (1.64) si ottiene

$$f_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n^d}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right). \tag{1.68}$$

L'equazione (1.68) esprime una stima di f che indichiamo con f_n (o in maniera equivalente possiamo indicarla con $\hat{f}$) come media di funzioni di $\mathbf{x}$ e di campioni $\mathbf{x}_i$. Infatti, la funzione finestra φ indicante il *kernel*[8] rappresenta una funzione finestra usata per interpolare la distribuzione dei dati, ossia, ciascun campione contribuisce alla stima di f in accordo alla distanza da $\mathbf{x}$. Le funzioni kernel usate in [**?**] sono: il kernel di Epanechnikov e Gaussiano. Il kernel di Epanechnikov ha il profilo

$$k_E(x) = \begin{cases} 1-x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \tag{1.69}$$

e la sua versione multivariata in $\Re^d$è data da

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|x\|^2) & \|x\| \leq 1 \\ 0 & altrimenti \end{cases} \tag{1.70}$$

con c_d rappresentante il volume della sfera d-dimensionale[9]. Il profilo della Normale è dato da

$$k_N(x) = \exp\left(-\frac{1}{2}x\right) \quad x \geq 1 \tag{1.71}$$

e la sua versione multivariata in $\Re^d$ è data da

$$K_N(\mathbf{x}) = (2\pi)^{-d/2}\exp\left(-\frac{1}{2}\|x\|^2\right). \tag{1.72}$$

Quindi riscrivendo la (1.68) abbiamo

$$\hat{f}(\mathbf{x}) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right) \tag{1.73}$$

che è rappresentata in figura 1.4, in cui è ricostruita una stima della densità f per una distribuzione bimodale. Siamo interessati a ricercare le mode della distribuzione f, e pertanto, valuteremo dove il gradiente ∇f si annulla, quindi:

$$\hat{\nabla} f(\mathbf{x}) = \nabla \hat{f}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}-\mathbf{x}_i)k'\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right). \tag{1.74}$$

Definiamo la funzione $g(x) = -k'(x)$ che risulta essere profilo per il kernel $G(\mathbf{x}) = c_{g,d}g\left(\|\mathbf{x}\|^2\right)$, con $c_{g,d}$ la costante di normalizzazione. Il kernel $K(\mathbf{x})$ è chiamato l'ombra di $G(\mathbf{x})$, che per Epanechnikov l'ombra è un kernel uniforme[10], mentre per la normale è sempre una normale. Quindi riscivendo la (1.74) abbiamo:

$$\begin{aligned} \hat{\nabla} f(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\underbrace{\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x}\right]}_{\mathbf{m}_{h,G}(\mathbf{x})} \end{aligned} \tag{1.75}$$

con il primo termine proporzionale alla stima della densità $\hat{f}$ in $\mathbf{x}$ calcolata con il kernel G, mentre il secondo termine è il vettore del mean shift $\mathbf{m}_{h,G}(\mathbf{x})$, che è la differenza tra la media pesata

[8] Da qui il nome di *kernel density estimation* (KDE). Spesso tali funzioni kernel di base (indicante un profilo) sono indicate con la lettera minuscola $k(\cdot)$.

[9] Nei nostri esperimenti $c_d = 1$.

[10] In questo caso $g(x) = 1$.

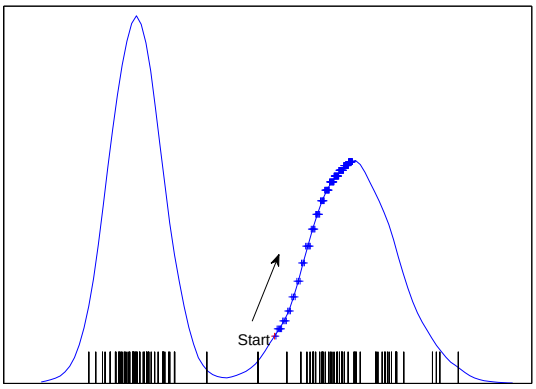


Figura 1.4: Rappresentazioen grafica nello spazio 1D del mean shift. Dato un insieme di punti generato da due distribuzioni Gaussiane (barrette nere), è stata approssimata la funzione densità con la (1.73) usando un kernel Gaussiano (1.72) con larghezza della finestra $h = 1$. Selezionando un punto qualsiasi dall'insieme dei campioni (croce rossa), il mean shift inizia a stimare la posizione successiva nella direzione in cui f aumenta, fino ad arrivare al valore in cui il gradiente si annulla. Si noti che per trovare tutte le mode, i campioni visitati devono essere eliminati ed una nuova procedura di ricerca della moda va inizializzata con un campione casuale mai visitato. Il mean shift termina quando tutti i campioni sono stati visitati. Si noti che nel grafico la densità f è stata calcolata per una migliore comprensione del funzionamento del mean shift. In situazioni reali, la densità f è non nota.

(usando il kernel G come pesi) ed il vettore $\mathbf{x}$ centro del kernel (finestra). Quindi la procedura del mean shift si può riassumere come segue

Algorithm 4 Algoritmo Mean shift per un kernel Normale

Siano dati i campioni da cui calcolare le mode $\mathbf{x}_i$ per $i = 1, \ldots, n$,
repeat
  $\mathbf{y}_{old} \leftarrow random(\mathbf{x}_i)$ Seleziona casualmente un punto iniziale dal dataset
  repeat
    Calcola il successivo valore centrale del kernel

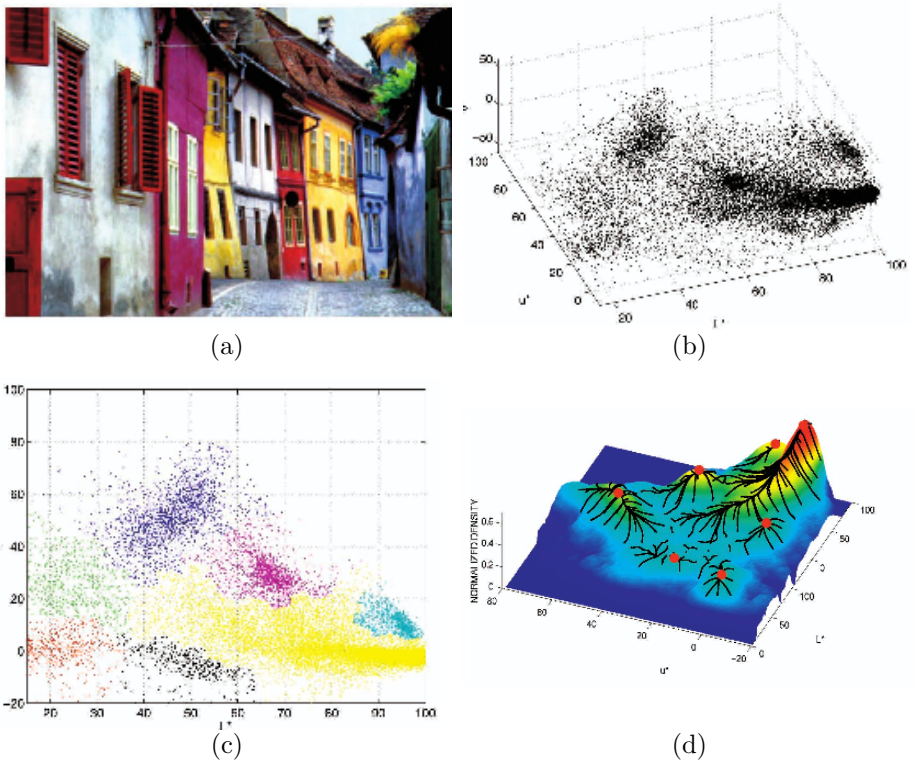
$$\mathbf{y}_{new} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \exp\left(\| \frac{\mathbf{y}_{old} - \mathbf{x}_i}{h} \|^2\right)}{\sum_{i=1}^{n} \exp\left(\| \frac{\mathbf{y}_{old} - \mathbf{x}_i}{h} \|^2\right)}$$

    if $\|\mathbf{y}_{new} - \mathbf{y}_{old}\| < \epsilon$ **then**
      break
    else
      $\mathbf{y}_{old} = \mathbf{y}_{new}$
    end if
  until $\|\mathbf{y}_{new} - \mathbf{y}_{old}\| < \epsilon$
until sono stati visitati tutti i campioni $\mathbf{x}_i$

Condizione sufficiente per la convergenza dell'algoritmo è dato dal seguente teorema

Teorema 1.5.1 *Se il kernel K ha un profilo convesso e monotonicamente decrescente, le sequenze di nuovi centri del kernel $\{\mathbf{y}_j\}_{j=1,2,\ldots}$ ed i rispettivi valori della funzione densità $\left\{\hat{f}(\mathbf{y}_j)\right\}_{j=1,2,\ldots}$ convergono. La sequenza $\left\{\hat{f}(\mathbf{y}_j)\right\}_{j=1,2,\ldots}$ è monotonicamente crescente.*

La dimostrazione di tale teorema è presente in [**?**]. Tra i due kernel menzionati, quello che produce un andamento più livellato è quello Gaussiano, a discapito del costo computazionale. Si può



(a) (b) (c) (d)

Figura 1.5: Immagine 400×276 in (a) e corrispondenti punti nello spazio L*u*v* (b). In (c) il risultato del clustering con il mean shift ed in (d) la sua densità e le mode trovate (da [**?**]).

segmentare una immagine a colori usando informazione congiunta tra colore e posizione del pixel nell'immagine. In particolare, l'informazione del pixel $\mathbf{x}_s = (x, y)$ che è indicato come dominio spaziale, è concatenata con quella di colore $\mathbf{x}_r$ che spesso è una terna di uno spazio di colore qualsiasi. L'algoritmo del mean shift è quindi applicato al dominio 5D. Poichè vi possono essere differenze in scala tra i domini di colore e spaziale, il kernel risultante è modificato in maniera opportuna:

$$K(\mathbf{x}_j) = k\left(\frac{\|\mathbf{x}_r\|^2}{h_r^2}\right) k\left(\frac{\|\mathbf{x}_s\|^2}{h_s^2}\right) \tag{1.76}$$

in cui vengono usati due raggi differenti h_r e h_s per controllare il comportamento del filtro composto. In figura 1.6 si riporta il risultato della segmentazione col mean shift con parametri $(h_r, h_s, M) = (16, 19, 40)$ in cui le regioni (componenti connesse dopo la segmentazione) che hanno un numero di pixel inferiore a M vengono eliminate.

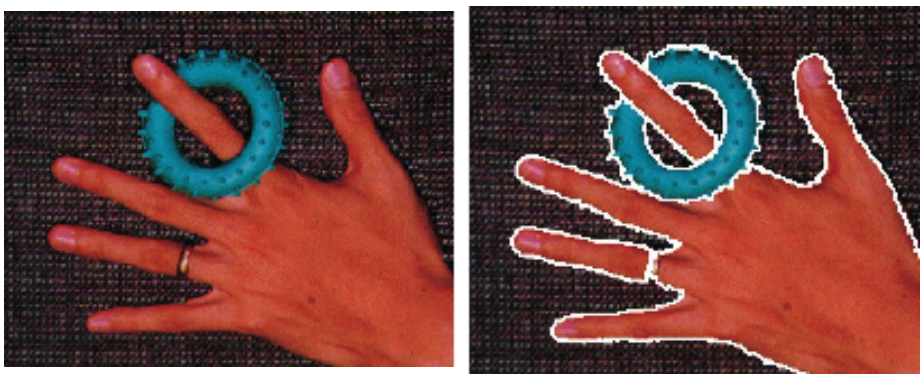


Figura 1.6: Immagine originale a sinistra e segmentata a destra con $(h_r, h_s, M) = (16, 19, 40)$(da [**?**]).