Università degli Studi di Milano - Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea in Informatica Magistrale

Fuzzy Inspirations in Unsupervised Learning

Oltolini Edoardo
Matricola: 869124

Contents

1 Background

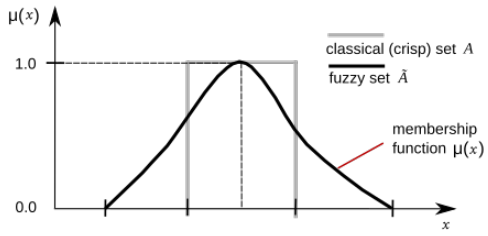
1.1 Argomenti relativi agli Insiemi

1.1.1 Funzione Caratteristica di un Insieme

Sia A $\subseteq$ X, la funzione caratteristica di A è la funzione da X in {0,1} che, sull'elemento x $\in$ X, vale 1 se x $\in$ A, 0 altrimenti.[13]

1.1.2 Membership Function

Per ogni insieme X, la membership function su X è una funzione da X in [0,1]. E' una generalizzazione della funzione caratteristica. Le funzioni di membership rappresentano sottoinsiemi fuzzy di X.[14]



1.2 Fuzzy Set e Teoria della Possibilità

Se consideriamo la frase: "Tizio è QUASI calvo", ci stiamo domandando se appartiene alle persone calve. Questo concetto è associato ai Membership Degree. Mentre se diciamo: "Tizio è quasi certamente calvo", allora l'incertezza è riferita a "certamente"; non sono sicuro che sia calvo, ma se lo è, appartiene completamente all'insieme di persone calve. Quest'ultima frase ci permette di capire la Teoria della Possibilità.

Nella Teoria della Possibilità i fuzzy set prendono il nome di Distribuzioni di Possibilità. Sono delle funzioni π : S $\rightarrow$ [0,1], dove S è l'universo del discorso e π(x) rappresenta lo stato di conoscenza che un certo agente ha su questo elemento x $\in$ S.

Esempio: se π(x) = 1, x è totalmente possibile. Se π(x) = 0, x è impossibile. **Esempio**: sappiamo che un'automobile è scura, allora è possibile che il colore sia blu, grigio, nero... è impossibile il colore bianco. Se abbiamo delle informazioni è possibile restringere il campo delle possibilità.

Totale ignoranza: $\forall$x $\in$ S, π(x) = 1, tutti gli stati sono ugualmente e totalmente possibili. **Conoscenza completa:** $\exists$x $\in$ S, π(x) = 1 e $\forall$b $\neq$ x, π(b) = 0.

Tra conoscenza completa e totale ignoranza ci sono tantissime sfumature ed è quindi possibile avere distribuzioni di possibilità **normalizzate**: $\exists$x t.c. π(x) = 1.

La misura di Possibilità ha un suo duale, la misura di Necessità:
Misura di Possibilità: $\Pi(A) = \max_{a \in A} \{\pi(a)\}$.
Misura di Necessità: $N(A) = 1 - \Pi(A^C) = \min_{a \notin A} \{1 - \pi(a)\}$

Si dimostra che: $N(A) \leq \Pi(A)$, se una cosa è necessaria, è anche possibile.

Alcune proprietà sono:

- $\Pi(A \cup B) = \max\{\Pi(A), \Pi(B)\}$
- $N(A \cap B) = \min\{N(A), N(B)\}$
- $N(A \cup B) \geq \max\{N(A), N(B)\}$, perché potrei avere che l'unione mi da un valore più alto di necessità.
- $\Pi(A \cap B) \leq \min\{\Pi(A), \Pi(B)\}$, discorso analogo

Probabilità	**Possibilità**
$\sum p(a) = 1$	$\exists a, \pi(a) = 1$
$P(A) = \sum_a p(a)$	$\Pi(A) = \max\{\pi(a)\}$
$P(A) = 1 - P(A^C)$, è auto duale	$\Pi(A) = 1 - N(A^c)$, non è auto duale
$P(A) + P(A^C) = 1$	$\Pi(A) + \Pi(A^c) \geq 1$

Table 1: Confronto tra probabilità e possibilità.

1.2.1 α-Level

Un Fuzzy Subset A di U è formato da elementi di U con un ordine gerarchico dato dai gradi di appartenenza. Un elemento x di U sarà in una "order class" α se il suo "membership value" è $\geq \alpha$, con $\alpha \in [0, 1]$, reale.

Formalmente un **α-Level** è un insieme classico definito come:

$$[A]^\alpha = \{x \in U \ : \ \varphi_A(x) \geq \alpha\}, \ con \ 0 \leq \alpha \leq 1, \ e \ \varphi_A : membership \ function$$

1.2.2 Fuzzy Numbers

Un sottoinsieme fuzzy A si dice che è un numero fuzzy quando l'Universo su cui φ_A è definita è l'insieme di tutti i numeri reali e soddisfa le seguenti condizioni: [6]

(i) all the α-levels of A are not empty for $0 \leq \alpha \leq 1$;
(ii) all the α-levels of A are closed intervals of $\mathbb{R}$;
(iii) supp $A = \{x \in \mathbb{R} : \varphi_A(x) > 0\}$ is bounded.

Rappresentiamo gli α-level di un numero fuzzy A come: $[A]^\alpha = [{a_1}^\alpha, {a_2}^\alpha]$. Ogni numero reale r è un numero fuzzy la cui funzione di membership è la funzione caratteristica χ_r :

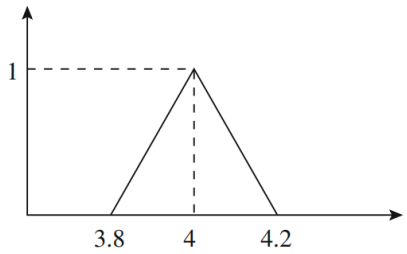
$$\chi_r(x) = \begin{cases} 1 & \text{if } x = r \\ 0 & \text{if } x \neq r \end{cases}.$$

L'insieme di tutti i numeri fuzzy si denota con $F(\mathbb{R})$. Di solito abbiamo a che fare con numeri fuzzy "triangolari", "trapezoidali" e "Gaussiani".

Un numero fuzzy A è "triangolare" se la sua funzione di membership è:

$$\varphi_A(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{u-a} & \text{if } a < x \leq u \\ \frac{x-b}{u-b} & \text{if } u < x \leq b \\ 0 & \text{if } x \geq b \end{cases},$$

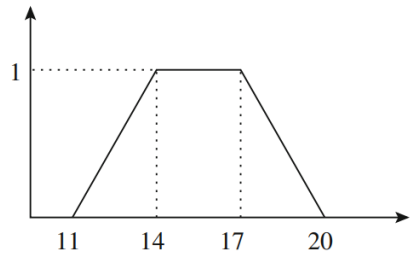
Esempio: (L'intervallo non deve essere per forza simmetrico)



E' detto "trapezoidale" se la sua funzione di membership è data da:

$$\varphi_A(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } a \leq x < b \\ 1 & \text{if } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{if } c < x \leq d \\ 0 & \text{otherwise} \end{cases},$$

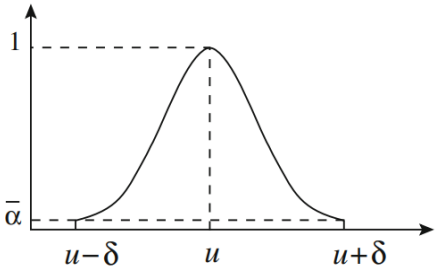
Esempio:



Un numero fuzzy è "bell-shaped" se la funzione di membership è "smooth" e simmetrica in relazione ad un numero reale.

$$\varphi_A(x) = \begin{cases} \exp\left(-\left(\frac{x-u}{a}\right)^2\right) & \text{if } u-\delta \le x \le u+\delta \\ 0 & \text{otherwise} \end{cases}.$$

Graficamente:



dove: $\overline{\alpha} = e^{-\left(\frac{\delta}{\alpha}\right)^2}$

1.3 Argomenti relativi alla Statistica

1.3.1 Covarianza

La covarianza di due variabili statistiche X e Y, σ_{XY} = Cov(X, Y) è un indice di variabilità congiunta. Date N osservazioni congiunte (xi, yi), di rispettive medie $\overline{x}$ e $\overline{y}$, la covarianza è [12]:

$$\sigma_{X,Y} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{N}\sum_{i=1}^{N} x_i y_i - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)\left(\frac{1}{N}\sum_{i=1}^{N} y_i\right).$$

1.3.2 Principal Component Analysis

La base matematica della PCA risiede nel calcolo degli autovalori λ_i e autovettori e_i di una matrice M, quindi tali che: M * e_i = λ_i * Id * e_i. Un altro concetto importante è il calcolo della matrice di covarianza. Le componenti principali sono combinazioni lineari delle variabili originali nella forma:

$$PC = a_{i1}X_1 + a_{i2}X_2 + ... + a_{ip}X_p$$

Le Xi sono le variabili, le a_{ij} sono gli elementi dell'autovettore e_i. Un primo vincolo da introdurre è che $a_{i1}^2 + \ldots + a_{ip}^2 = 1$, questo viene fatto in modo che Var(PC_i) non possa essere aumentata semplicemente incrementando uno degli a_{ij}. Ora, siccome i vettori e_1, ..., e_n sono ortonormali, si osserva che:

$$\mathbf{e}_i^{\mathrm{T}}\mathbf{M}\mathbf{e}_i = \lambda_i, \quad \mathbf{e}_i\mathbf{M}\mathbf{e}_j = 0 \quad \text{for } i \neq j$$

e che:

$$\mathbf{M} = \lambda_1 \mathbf{e}_1{}^{\mathrm{T}} \mathbf{e}_1 + \lambda_2 \mathbf{e}_2{}^{\mathrm{T}} \mathbf{e}_2 + \ldots + \lambda_n \mathbf{e}_p{}^{\mathrm{T}} \mathbf{e}_p$$

Questa espressione è detta "decomposizione spettrale di M". La proprietà di queste nuove variabili è la mancanza di correlazione tra esse. La varianza dell'i-esimo componente è:

$$\mathrm{Var}(\mathbf{e}_i X) = \lambda_i$$

Dove:

$$\mathrm{Cov}(\mathbf{e}_i X, \mathbf{e}_j X) = 0 \quad \text{for } i \neq j$$

La prima componente principale (PC1) è la combinazione lineare dei valori dei campioni per cui i "punteggi" hanno la massima varianza. La PC2 ha "punteggi" che non sono correlati e con quelli per la PC1. La PC3 è la combinazione lineare che ha la massima varianza su tutte le combinazioni i cui punteggi non sono correlati con quelli delle prime due componenti, e così via. Quindi viene fatta una trasformazione degli assi principali della matrice di covarianza dei dati, selezionando gli autovettori corrispondenti agli autovalori più grandi, fino a che la somma di questi non sia almeno una frazione specificata dall'utente della somma di tutti gli autovalori, che rappresenta la frazione di varianza preservata. Tuttavia la PCA è suscettibile a valori mancanti, imprecisi e outlier.[5]

1.4 Argomenti relativi al Machine Learning

1.4.1 Algoritmo k-Means

L'obiettivo è determinare k gruppi di dati, assumendo che gli attributi formino uno spazio vettoriale. Si punta anche a minimizzare la distanza intra-cluster e massimizzare quella inter-cluster.

Pseudocodice:
1. Inizializza casualmente i centroidi {c1, c2, ..., ck}.

2. Ripeti:
a. Assegna xi al cluster P_j se $d(xi, cj)^2$ è minimo. **(distanza euclidea)**
b. Per ogni cluster P_i, calcola il nuovo centroide c_i: $\mathbf{c_i}$ = media dei punti assegnati al cluster i

c. Controlla la convergenza (o massimo numero di iterazioni):
Se i centroidi non cambiano significativamente (variazione < tolleranza), termina.
Altrimenti, continua.

3. return (centroidi finali $\{c_1, c_2, \ldots, c_k\}$ **e i cluster** $\{P_1, P_2, \ldots, P_k\}$**).**

1.4.2 Misura di Silhouette

La tecnica di valutazione maggiormente utilizzata nei problemi di clustering è la Silhouette, una misura "interna". Non si utilizzano "ground truth" (dati etichettati) ma si punta a minimizzare la distanza intra-cluster e massimizzare quella inter-cluster. La Silhouette combina idee di coesione e separazione. Dato un punto individuale i:

- Calcola a = distanza media di i dai punti nello stesso cluster
- Calcola b = min(distanza media di i dai punti in altri cluster)
- Coefficiente di Silhouette: s_i = (b – a) / max(a, b)
  - Il valore può variare tra -1 e 1 (tipicamente tra 0 e 1)
  - Più è vicino a 1, migliori sono i risultati
  - Se è = -1, l'oggetto è stato inserito nel cluster sbagliato
  - Se è = 0, l'oggetto è nella "boundary region"

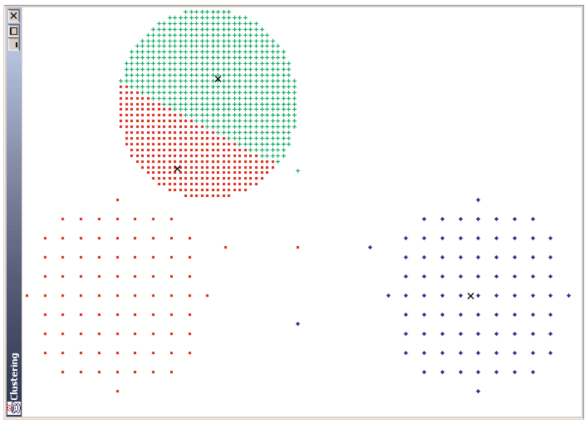
Lo score medio di silhouette su tutti gli oggetti, spesso usato come misura di qualità del clustering, è dato da [3]:

$$CS = \frac{1}{n}\sum_{j=1}^{n} s_j$$

1.4.3 Fuzzy C-Means (FCM)

Ogni cluster C_i è rappresentato da un fuzzy set f_{Ci}. Il primo vincolo è che: $\sum_i f_{C_i}(o_j) = 1$, cioè la somma dei gradi di appartenenza di o_j ai vari cluster deve essere uguale a 1. Gli f_{Ci} sono fuzzy set ma il vincolo deriva più dalla teoria delle probabilità. Si assegna l'oggetto al cluster che è più vicino, ma lo si fa con un grado di membership basato sulla distanza rispetto ai cluster.

Vantaggi (w.r.t. k-Means): cerca di evitare minimi locali, vuole raggiungere quello globale. **Problemi:** può generare minimi globali inesistenti quando un cluster è molto più denso di un altro e, se non gestito, gli outlier apparterranno per forza ad almeno uno dei cluster.



Funzione obiettivo (**S**um of **S**quared **E**rrors):

$$SSE_f = \sum_{i=1}^{C} \sum_{k=1}^{N} u_{ki}^{w} * d(x_k, v_i)^2$$

Dove:

- u^w_{ij} è una funzione convessa in [0, 1].
- $w > 1$, è detto "Fuzzifier", esprime l'impatto dell'aggiunta di u_{ki} al K-Means standard, solitamente è uguale a 2, così il Fuzzy Set somiglia ad una funzione Gaussiana.
- $u_{ki} \in [0,1]$ è il grado di appartenenza di x_k a C_i (quindi è il fattore che mi va a definire i fuzzy set dei cluster)
- $\sum_{j=1}^{c} u_{kj} = 1$, è il vincolo detto in precedenza, $\forall$k, t.c.: $1 \leq k \leq N$
- v_i, sono i centri dei cluster

Se w = 1 oppure $u_{ki} \in \{0,1\}$ è congruo al K-Means. Solitamente la funzione di distanza è la distanza Euclidea. Algoritmo:

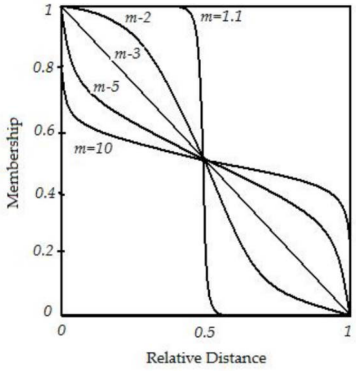
- **Inizializzazione:** genero casualmente gli insiemi fuzzy (cioè i gradi/pesi "u")
- **Iterazione:**
  - Aggiorno i centroidi per minimizzare SSE_f
  - Aggiorno i pesi per minimizzare SSE_f
- **Ripeto** fino al criterio di **convergenza** (cioè non diminuisce più l'errore) oppure ho superato un numero di iterazioni prefissato dall'utente.

Per l'aggiornamento si usano le seguenti due formule:

$$u_{ki} = \frac{\frac{1}{D(x_k, v_i)^{\frac{1}{w-1}}}}{\sum_{j=1}^{C} \frac{1}{D(x_k, v_j)^{\frac{1}{w-1}}}}$$

$$v_i = \frac{\sum_{k=1}^{N} (u_{ki})^w * x_k}{\sum_{k=1}^{N} (u_{ki})^w}$$

Il valore di m che si usa più spesso è 2, questo perchè come nell'immagine seguente, i fuzzy set diventano "bell-shaped".



Si noti che se m = 1.1, i fuzzy set tendono ad essere simili a insiemi classici (i gradi di appartenenza sono circa {0,1}). Con m > 2 si formano dei picchi in cui si vanno a concentrare i valori di fuzziness.

2 Fuzzy Sets in Data Analysis

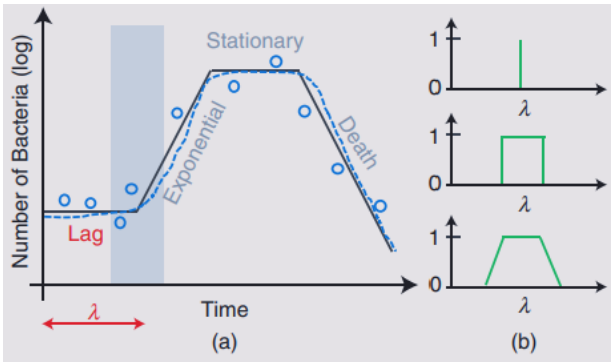
2.1 Introduzione

I Fuzzy Set furono ideati nel '65 da Zadeh, e da lì in poi sono stati utilizzati in numerosi campi come la Control Theory, la Ricerca Operativa, l'Ottimizzazione e l'Information Retrieval. In questo approfondimento si vuole mostrare come l'espressività dei Fuzzy Set (basata sul concetto di "Graded Truth") può contribuire all'analisi dei dati e al Machine Learning. Lo stesso Zadeh, nel '65, prevede che i Fuzzy Set saranno applicati in campi come la Pattern Classification e l'Information Processing. Nonostante la lungimiranza, nei primi 30 anni, ci si è focalizzati sulla "Control Theory" ed il "Reasoning" approssimato. Il focus è cambiato di recente e ciò è dipeso da fattori come la nascita della "Data Science" e l'attuale dominanza in letteratura del Machine Learning.[4]

2.2 Fuzzy Data and Statistical Concepts

Nella Data Analysis, di solito si assume che i dati siano stati ottenuti da misurazioni precise. Consideriamo però il seguente esempio biologico: una popolazione di batteri ha

4 fasi: “Lag Phase”, “Exponential (Growth) Phase”, “Stationary Phase”, “Death Phase”. La curva, che rappresenta il numero di batteri nel tempo, è:



La curva, tratteggiata, ha transizioni smussate e una serie di rumori dovuti al dispositivo. Supponiamo di voler stimare il “Lag Time” λ. Come determino precisamente λ, se dipende anche dalle condizioni esterne, come la temperatura? Una possibile soluzione, per stimarla precisamente, è utilizzare le distribuzioni di probabilità. Ma se non so il valore reale, come conosco la sua probabilità?

Per affrontare questo problema si può utilizzare un’analisi fuzzy. Si può modellare (b) λ con un Fuzzy Set trapezoidale, come un intervallo (caso particolare di Fuzzy Set), oppure come un numero preciso (caso particolare di intervallo).

In letteratura l’analisi dei dati fuzzy ha seguito 2 linee parallele: l’interpretazione ”Epistemica” e quella ”Ontica” dei fuzzy set. Dal punto di vista epistemico, i fuzzy set informazioni incomplete sugli elementi di un universo X, ed i valori di “membership” sono dei gradi di **possibilità:** se la nostra conoscenza su un oggetto sconosciuto $x_0 \in X$ è modellata da un Fuzzy Set $\widetilde{A}$, allora $\forall x \in X$, il valore di membership $\widetilde{A}(x)$ è il grado di possibilità che $x_0 = x$.

Dal punto di vista “Ontico”, i fuzzy set rappresentano informazioni precise su entità complesse, cioè punti nello spazio F(X) dei Fuzzy Subset di X. Ad esempio, possiamo dire che il “Lag Time” è una quantità Fuzzy che può assumere un valore che appartiene in qualche misura a diversi Fuzzy Subset: breve, medio, lungo.[4]

2.3 Data Pre-Processing

Ci concentriamo sulla Fuzzy PCA, sezione 2.4 e sul rilevamento di outlier 2.5.

2.4 Fuzzy PCA

La PCA ha lo scopo di ridurre la dimensionalità di un dataset mappandolo in uno spazio lineare a bassa dimensione, preservando la maggior parte della varianza dei dati. Nella

Fuzzy PCA la prima componente principale viene trovata come soluzione all'algoritmo "fuzzy 1-line" (variante dell'FCM con prototipi lineari monodimensionali e c = 1) esteso con un Noise Cluster. Infatti se il prototipo del cluster è una retta, al posto di un punto, i cluster avranno una forma a "tubo" e così via.

Il prototipo, la prima componente principale, viene denotata con L(u,v), dove v è il centro della classe e u, con $\|u\| = 1$, che è la direzione principale, associata all'autovalore di modulo massimo λ_{max} della matrice di covarianza fuzzy:

$$C_{kl} = \frac{\sum_{j=1}^{n} [A_i(x^j)]^2 (x_{jk} - \bar{x}_k)(x_{jl} - \bar{x}_l)}{\sum_{j=1}^{n} [A_i(x^j)]^2}$$

Dove $A_i(x^j) \in [0,1]$ rappresentano i membership degree del punto x^j al cluster A_i, mentre x_{jk} è la k-esima componente del punto x^j e infine $\bar{x_k}$ è la media pesata della componente k-esima su ogni membership degree. L'algoritmo di FPCA proposto in [5] determina i valori $A(x^j)$ che descrivono meglio il Fuzzy Set A e la relazione con il suo prototipo lineare (prima componente principale).

$$A(x^j) = \frac{\alpha/(1-\alpha)}{[\alpha/(1-\alpha)] + d^2(x^j, L)}$$

α rappresenta il membership degree del punto più distante (l'outlier più grande) della prima componente principale. Siccome è un parametro di input, serve un'euristica per determinarlo. Siamo interessati a trovare i membership degree che contribuiscono nella produzione della miglior prima componente principale per il data set.

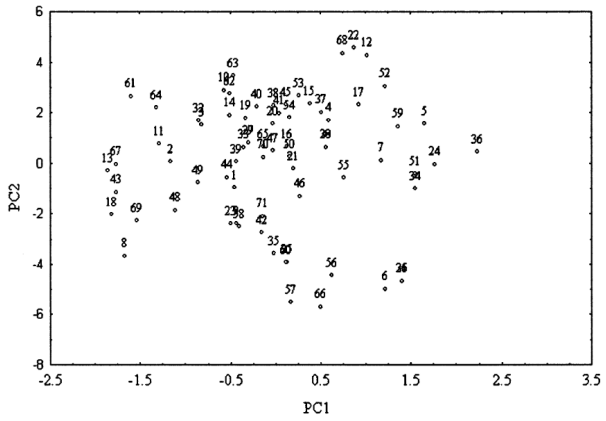
La funzione obiettivo da minimazzare è:

$$J(A, L; \alpha) = \sum_{j=1}^{n} [A(x^j)]^2 d^2(x^j, L) + \sum_{j=1}^{n} [\bar{A}(x^j)]^2 \frac{\alpha}{1-\alpha}$$

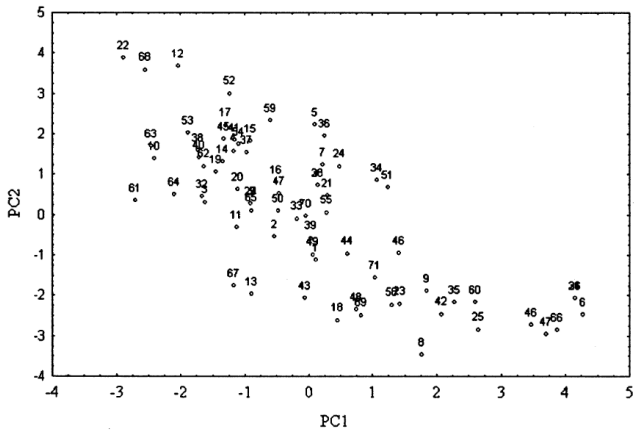
Siccome l'autovalore associato ad una componente principale descrive quanto i dati sono dispersi lungo quella componente, vogliamo che la prima componente sia con l'autovalore più grande possibile. Allora α dovrà massimizzare quell'autovalore, ma il valore esatto non è necessario; si seleziona l'α che permette di avere un compromesso tra minimizzare la funzione e l'ottenere l'autovalore di modulo massimo.

Usando i fuzzy membership degree ottenuti, si ricalcola la matrice di covarianza e si determinano i suoi autovalori e autovettori, che sono le componenti principali fuzzy e i corrispondenti valori di dispersione.

Gli autori propongono la FPCA per una stima robusta delle componenti principali. L'efficienza è stata illustrata su un set di dati riguardante l'interazione dei legami carbonio-idrogeno con i legami molibdeno-osso, ottenendo risultati migliori rispetto alla PCA classica. Usando 2 componenti, l'FPCA spiega il 97,20% della varianza totale, la PCA solo il 69,75%. Quindi la FPCA rende l'analisi del set di dati molto più semplice. Inoltre, la PCA ha mostrato solo una separazione parziale dei complessi chimici sul piano descritto dalle prime due componenti principali.



Mentre si osserva una differenziazione molto più netta delle variabili metriche lungo la diagonale con l'FPCA.



Questi risultati dovrebbero incoraggiare l'applicazione della metodologia dell'analisi delle componenti principali fuzzy ad altri sforzi di "data mining".

2.5 Outlier Detection - Noise Clustering

Un outlier è un particolare valore che, in qualche modo, non è come gli altri data point, ma è un rumore (un punto molto distante dagli altri) [4]. Per gestire gli outlier è stata creata una variante dell'FCM, esteso con un **Noise Cluster**.

L'idea del Noise Clustering, proposta da Davé, è aggiungere un cluster 0, la cui membership è u_{0k}, $\forall$k = 1, ..., n, con la speranza che raccolga gli outlier.

Il rumore è quindi una classe separata, rappresentata da un centroide extra, che ha distanza costante δ, da tutti i vettori "feature". La funzione da minimizzare diventa [11]:

$$J_2(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m D_{ik} + \sum_{k=1}^{n}(u_{0k})^m \delta^2$$

2.6 Possibilistic Clustering

Uno dei metodi fuzzy più popolari per l'analisi dei dati è il Fuzzy Clustering. Oltre alle varianti già citate dell'FCM, un'altra è il Clustering Possibilistico.[4]

- Rilassa il vincolo sui gradi di membership. La somma ora può essere $\leq$ o ≥ 1. Almeno un cluster deve contenere l'oggetto o_j, quindi sicuramente la somma è > 0.
- Deve essere modificata la funzione obiettivo, altrimenti avremo sempre $u_{ki} = 0$
- Le f_{C_i} diventano distribuzioni possibilistiche, $f_{C_i}(o_j)$ rappresenta la **tipicality** dell'oggetto o_j nel cluster Ci
- E' sensibile alle condizioni iniziali (come il k-Means), ritorna il problema dei minimi locali, motivo per cui l'FCM rimane più utilizzato.

Nella versione di Krishnapuram e Keller, è formulato come:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{H}} J_{PkM} &= \sum_{i=1}^{n}\sum_{g=1}^{k} u_{ig}^m d^2(\mathbf{x}_i, \mathbf{h}_g) + \sum_{g=1}^{k}\eta_g \sum_{i=1}^{n}(1-u_{ig})^m, \\ \text{s.t.} \quad & u_{ig} \in [0,1], i = 1,\ldots,n, \forall g = 1,\ldots,k, \end{aligned}$$

Dove η_g è un parametro di "tuning" associato al cluster g, che pesa il suo contributo nella funzione di "penalizzazione". La funzione di perdita contiene un termine aggiuntivo, che gioca un ruolo nell'evitare soluzioni triviali con $u_{ig} = 0$, $\forall$i = 1, ..., n, $\forall$g = 1, ..., k. Ovviamente rimuove il vincolo per cui la somma sulle righe di U è 1. **Aggiornamenti:**

- Per i prototipi dei cluster possiamo usare la formula per i centroidi v_i descritta in sezione 1.4.3, con i valori che qui sono chiamati h_g.
- I gradi di membership vengono aggiornati in questo modo:

$$u_{ig} = \frac{1}{1 + \left(\frac{d_{ig}^2}{\eta_g}\right)^{\frac{1}{m-1}}}.$$

- I coefficienti η, vengono calcolati come:

$$\eta_i = K \frac{\sum_{j=1}^{N} u_{ij}^m d_{ij}^2}{\sum_{j=1}^{N} u_{ij}^m}.$$

  dove K è solitamente uguale a 1.

Gli autori presentano questo approccio sostenendo che i metodi di clustering fuzzy non forniscano valori di appartenenza adeguati per applicazioni in cui le appartenenze devono essere interpretate come gradi di compatibilità o possibilità. Ciò è dovuto al fatto che tali metodi utilizzano un vincolo probabilistico, di conseguenza, l'appartenenza di un vettore a un cluster dipende non solo dalla sua posizione, ma anche dalla distanza rispetto agli altri cluster.

Ciò impone che i valori di appartenenza siano distribuiti tra le classi, rendendoli dipendenti dal numero di cluster presenti. I valori di membership risultanti non sempre riescono a distinguere tra buoni e cattivi membri. Questa situazione si verifica perché i gradi probabilistici non sono in grado di distinguere tra "ugualmente probabile" e "sconosciuto".

Adottando una prospettiva possibilistica, in cui l'appartenenza di un punto ad una classe è indipendente dalla sua appartenenza ad altre classi, possiamo generare distribuzioni di appartenenza che modellano l'indeterminatezza [7].

Il Clustering Possibilistico soffre tuttavia del "coincident clusters problem": la soluzione è solitamente formata da un unico cluster, poiché il problema di ottimizzazione si può ridurre in una somma di k problemi di minimizzazione (uno per ogni cluster), minimizzabili in modo indipendente. Un rimedio è partire dai centri dell'FCM per inizializzare il punto di partenza della versione possibilistica.

3 Altre Ispirazioni Fuzzy

Nell'Unsupervised Learning, l'algoritmo di apprendimento riceve un dataset ed i data point sono visti come vettori di attributi in uno spazio tipicamente euclideo. L'obiettivo è scoprire la loro distribuzione, le relazioni o le dipendenze.

3.1 Silhouette in FCM

La Silhouette classica serve per valutare solamente gli hard clustering. Ciò porta Campello e Hruschka (2006) a sviluppare la "Fuzzy Silhouette", che premia gli oggetti vicini ai centri dei cluster, riducendo l'importanza degli oggetti nelle regioni "boundary", in cui i

membership degree sono simili o identici tra cluster diversi. Denotando con p(j) e q(j) gli indici dei due cluster con i più alti valori di membership associati ad xj, e con s_j la silhouette classica, ottenuta con un processo di defuzzificazione, il punteggio è dato da [9]:

$$FS = \frac{\sum_{j=1}^{n}(u_{pj}-u_{qj})^{\alpha} s_j}{\sum_{j=1}^{n}(u_{pj}-u_{qj})^{\alpha}}$$

Dove la potenza α è solitamente uguale ad 1. Chiaramente, tale valutazione non è completa poiché tende a ignorare il clustering dei punti nelle regioni sovrapposte, ed è per questo che si cercano misure sempre più generalizzate.

3.2 Indice di Validità di Xie-Beni

L'indice di validità proposto da Xie e Beni [15] è uno strumento fondamentale per la valutazione della qualità di una partizione fuzzy ottenuta attraverso algoritmi di clustering fuzzy. Tale indice è progettato per identificare partizioni fuzzy che siano sia compatte che ben separate, superando le limitazioni delle metriche precedenti. In questo campo la validazione è più complessa rispetto al clustering hard. L'indice è calcolato come:

$$XB = \frac{\sum_{i=1}^{N}\sum_{k=1}^{c} u_{ik}^{m}\|x_i - v_k\|^2}{N \cdot \min_{j\neq k}\|v_j - v_k\|^2} \tag{1}$$

dove:

- N: numero di punti dati;
- c: numero di cluster;
- u_{ik}: grado di appartenenza del punto x_i al cluster k;
- m: parametro di fuzzificazione, che controlla il livello di "sfocatura" della partizione;
- $\|x_i - v_k\|$: distanza tra il punto x_i e il centroide v_k del cluster;
- $\min_{j\neq k}\|v_j - v_k\|$: distanza minima tra i centroidi di due cluster diversi.

L'indice combina due aspetti fondamentali:

1. **Compattezza**: il numeratore rappresenta la somma pesata delle distanze quadratiche tra i punti dati e i rispettivi centroidi, penalizzando cluster poco compatti;
2. **Separazione**: il denominatore rappresenta la minima distanza tra i centroidi di cluster diversi, penalizzando cluster vicini.

Valori più bassi di **XB** indicano una partizione fuzzy migliore. La sua capacità di bilanciare compattezza e separazione, insieme alla semplicità computazionale, lo rende un riferimento fondamentale in letteratura.

3.3 Fuzzy C-Means with Polynomial Fuzzifier

Idealmente, ogni trasformazione g(u) può essere scelta, basta che:

- g(0) = 0, g(1) = 1
- g deve essere derivabile e la derivata deve essere crescente

Siccome per aggiornare gli u_{ij} ci serve l'inversa della derivata prima di g, g', volendo rendere il fuzzifier polinomiale, un metodo semplice e computazionalmente efficiente è avere una trasformazione quadratica. La scelta si riduce a g(u) = α*u + (1 – α)*u, con $0 \leq \alpha \leq 1$. Al posto di α usiamo il parametro

$$\beta = \frac{g'(0)}{g'(1)} = \frac{1-\alpha}{1+\alpha}.$$

Possiamo calcolare facilmente $\alpha = \frac{1-\beta}{1+\beta}$. Assumiamo che d_{i0j} sia la distanza del punto xj dal cluster più vicino e che dij sia la distanza di xj da un altro cluster più distante. Allora β indica il limite inferiore che il quoziente $\frac{d_{i0j}}{d_{ij}}$ deve superare, in modo da non avere membership degree di xj che siano zero. Per $\beta = 0$ si ottiene un fuzzy clustering standard, con fuzzifier = 2 e per $\beta \to 1$, ci avviciniamo al crisp clustering. Ora la funzione obiettivo da minimizzare è:

$$f = \sum_{i=1}^{c}\sum_{j=1}^{n}\left(\frac{1-\beta}{1+\beta}u_{ij}^2 + \frac{2\beta}{1+\beta}u_{ij}\right)d_{ij}$$

Si ottiene, per $u_{ij} \neq 0$, l'aggiornamento:

$$u_{ij} = \frac{1}{1-\beta}\left(\frac{1+(\hat{c}-1)\beta}{\sum_{k:u_{kj}\neq 0}\frac{d_{ij}}{d_{kj}}} - \beta\right).$$

Bisogna però determinare quali $u_{ij} = 0$. Siccome vogliamo minimizzare la funzione obiettivo, Se $u_{ij} = 0$ e $d_{ij} < d_{tj}$, allora al minimo della funzione obiettivo, sicuramente avremo $u_{tj} = 0$, altrimenti potremmo ridurre il valore settando $u_{tj} = 0$ e facendo assumere ad u_{ij} il valore originario di u_{tj}.

Ciò implica che, per un indice j arbitrario, possiamo ordinare le distanze d_{ij} in ordine decrescente. Senza perdere di generalità, assumiamo che $d_{1j} \geq \ldots \geq d_{cj}$. Se non ci sono gradi di membership a zero, per minimizzare la funzione obiettivo, bisogna intervenire

sugli u_{ij} relativi a distanze elevate, che vanno posti a zero; la regola di aggiornamento non può essere applicata su questi u_{ij}.

Allora dobbiamo trovare l'indice più piccolo i_0 per cui l'aggiornamento descritto prima è applicabile, cioè per cui si ottiene un numero positivo. Per $i < i_0$ abbiamo $u_{ij} = 0$, mentre per $i \geq i_0$ il membership degree u_{ij} è calcolato come nella regola di aggiornamento, con $\widehat{c} = c + 1 - i_0$.

Si dimostra che $\beta = 0.5$ è generalmente un buon valore per il parametro. Rimane da definire l'equazione di aggiornamento per i prototipi dei cluster, ma questa rimane invariata rispetto all'FCM, con la sola differenza che al posto di $g(u_{ij}) = u_{ij}{}^m$, abbiamo [1]:

$$g(u_{ij}) \;=\; \alpha u_{ij}^2 + (1-\alpha)u_{ij} \;=\; \frac{1-\beta}{1+\beta}u_{ij}^2 + \frac{2\beta}{1+\beta}u_{ij}.$$

Klawonn e Hoppner propongono questo approccio cercando di superare il problema nel clustering fuzzy in cui i dati tendono a influenzare i cluster, come il fatto che dati densi trascinano i centroidi degli altri cluster. Dal punto di vista computazionale, l'algoritmo è leggermente più complesso. Le equazioni di aggiornamento sono simili all'FCM. Tuttavia, per ogni vettore di dati, dobbiamo ordinare le distanze rispetto ai cluster in ogni fase iterativa.

3.4 Possibilistic Clustering With Repulsion Constraints

Per risolvere il problema dei cluster coincidenti possiamo avviare prima l'algoritmo FCM, per trovare i centroidi, da inserire come parametro di inizializzazione per l'algoritmo PCM. In aggiunta, è possibile formalizzare una variante del clustering possibilistico, Il PkM-R, come [10]:

$$\begin{aligned}\min_{\mathbf{U},\mathbf{H}} J_{PkM-R} \;=\; & \sum_{i=1}^{n}\sum_{g=1}^{k} u_{ig}^m d^2\left(\mathbf{x}_i, \mathbf{h}_g\right) + \sum_{g=1}^{k}\eta_g \sum_{i=1}^{n}\left(1-u_{ig}\right)^m \\ & + \sum_{g=1}^{k}\gamma_g \sum_{g'=1, g'\neq g}^{k} \frac{1}{\xi d^2\left(\mathbf{h}_g, \mathbf{h}_{g'}\right)} \\ \text{s.t.} \quad & u_{ig} \in [0,1], i = 1, \ldots, n, \forall g = 1, \ldots, k,\end{aligned}$$

Dove η_g è un parametro che regola l'importanza di uno specifico cluster, γ_g incrementa l'importanza del termine di repulsione, e ξ dipende dalla distanza minimale che vogliamo accettare tra due cluster vicini. Tuttavia si osserva che un buon valore è $\xi = 1$.

Quando 2 cluster sono quasi coincidenti la distanza tra i loro due prototipi è circa = 0, rendendo la frazione nel termine di repulsione, tendente ad infinito. Se invece la distanza è grande, la loro repulsione è praticamente nulla. Si nota però che l'aggiornamento della posizione dei cluster diventa molto complesso.

4 Applicazione degli Algoritmi di Clustering

4.1 Iris Dataset

Per prima cosa sono stati applicati i seguenti algoritmi sul dataset Iris: FCM, KM, FCM-PF. I risultati sono i seguenti:

Metriche	FCM-PF	KM	FCM
Silhouette (Sil)	0.7357	0.5528	0.7320
Silhouette Fuzzy (SilF)	0.7578	–	0.8091
Xie-Beni (XB)	0.1543	0.1628	0.1369

Table 2: Risultati delle metriche per PF, KM e FCM.

Come si vede dalla tabella 2, la silhouette è massima per l'algoritmo FCM con Polynomial Fuzzifier, ed è molto simile a quella del classico FCM. Per l'indice di Xie-Beni invece, quelle dell'FCM-PF e del K-Means risultano molto simili, mentre il risultato migliore è ottenuto con l'FCM Standard.

In secondo luogo, si è cercato di applicare il Possiblistic C-Means sullo stesso dataset, ottenendo come silhouette classica 0.84, mentre per la versione fuzzy 0.95. Questi numeri non ci devono trarre in inganno, in realtà l'algoritmo PCM è riuscito soltanto a trovare 2 cluster, mentre il terzo cluster è coincidente con uno degli altri, questo porta l'indice Xie-Beni ad essere altissimo (≈ 7000). Valori simili sono stati ottenuti tramite il PCM "Repulsive", che richiede però un tempo di calcolo eccessivo.

Ora confrontiamo i risultati con quelli ottenuti dopo l'applicazione della FPCA. Si noti che la Fuzzy PCA, rispetto alla PCA standard, fa ricadere molta più varianza sulle prime componenti. Con n=2, la varianza spiegata è 1. Sono stati ottenuti i seguenti risultati:

Metriche	FCM-PF	KM	FCM
Silhouette (Sil)	0.8023	0.6056	0.6286
Silhouette con Fuzzy (SilF)	0.8061	–	0.7131
Xie-Beni (XB)	0.1564	0.1557	0.3534

Table 3: Risultati delle metriche per PF, KM e FCM. (Fuzzy-PCA)

Il K-Means è migliorato leggermente, mentre l'FCM Standard è peggiorato sotto tutti i criteri di valutazione. L'FCM-PF è migliorato in entrambi i punteggi di Silhouette ed è rimasto congruente a prima rispetto all'indice XB. L'FCM con Polynomial Fuzzifier è quindi una buona scelta per questo dataset.

Successivamente, è stato fatto lo stesso procedimento con la PCA classica, ottenendo scarsi risultati per questi 3 algoritmi.

Avendo diminuoto le componenti a 2, ci si chiede se è possibile applicare l'FCM Possibilistico in entrambe le versioni. La risposta è no per la versione Standard del PCM, continuando a trovare un cluster coincidente, mentre è positiva per il PCM "Repulsive". Sono stati ottenuti i seguenti risultati:

Metriche	Valore
Silhouette (Sil)	0.7380
Silhouette con Fuzzy (SilF)	0.7988
Xie-Beni (XB)	0.0741

Table 4: Risultati delle metriche per PCM-R. (Fuzzy-PCA)

Si nota che la silhouette è molto alta e l'indice XB è molto basso.

Da qui possiamo concluedere che il PCMR può rientrare negli algoritmi migliori per l'Iris Dataset, tuttavia non è computazionalmente efficiente come i precedenti.

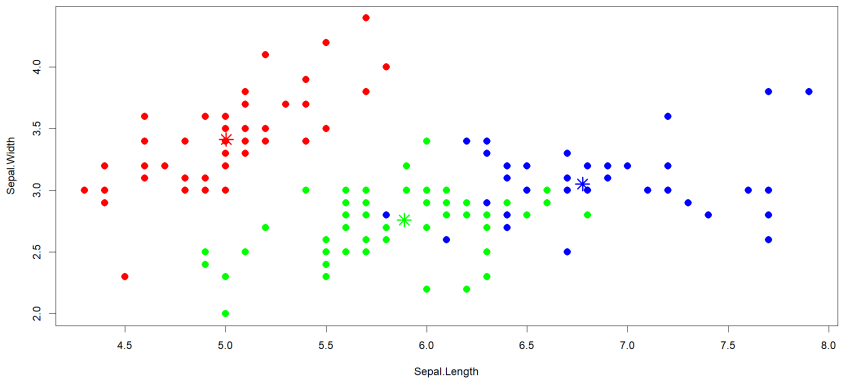


Figure 1: Esempio Plot Clustering su Iris Dataset

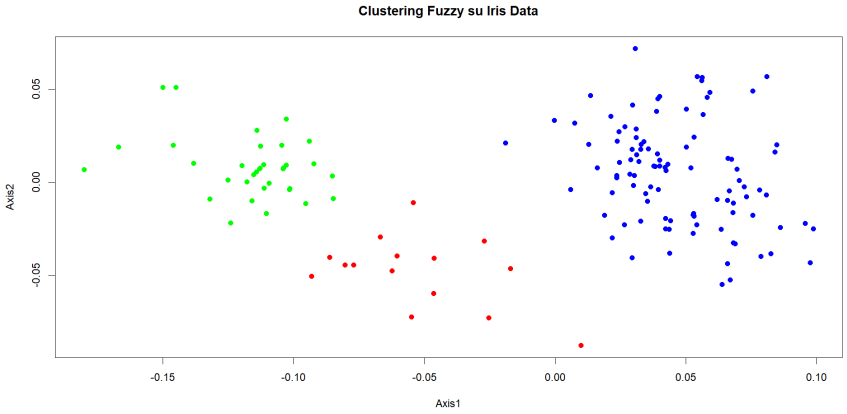


Figure 2: Iris + FPCA

4.2 Digits Dataset

Risultati ottenuti con KM, FCM, FCM-PF:

Metriche	FCM-PF	KM	FCM
Silhouette (Sil)	0.2984	0.1687	-0.0584
Silhouette con Fuzzy (SilF)	0.3755	–	0.0417
Xie-Beni (XB)	2.2441	1.6271	6.0807e+26

Table 5: Risultati delle metriche per PF, KM e FCM

L'FCM standard in questo caso è completamente inutilizzabile, mentre l'FCM-PF ha circa il doppio di punteggio di Silhouette media classica rispetto al KM, tuttavia ha un indice XB leggermente maggiore. Ciò può far pensare che l'FCM-PF sia un algoritmo robusto che si destreggia bene in molti casi.

Passando ai test con la Fuzzy-PCA, gli algoritmi possibilistici, con n = 4 componenti principali, sono stati scartati per il tempo di esecuzione. Il dataset non è del tutto adatto a problemi di Clustering ed ha uno shape di 1797x64. I risultati, con n = 4, sono:

Metriche	FCM-PF	KM	FCM
Silhouette (Sil)	0.4780	0.2969	0.4212
Silhouette con Fuzzy (SilF)	0.5385	–	0.5835
Xie-Beni (XB)	0.4129	0.5439	0.5183

Table 6: Risultati delle metriche per PF, KM e FCM. (Fuzzy-PCA)

Ora i risultati con n = 2.

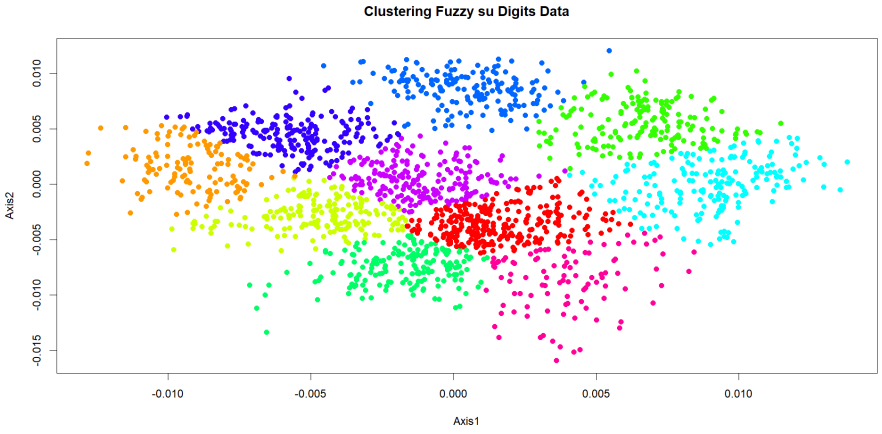


Figure 3: Digits, n = 2, k = 10, FCM

Anche in questi casi la Fuzzy PCA migliora i risultati ottenuti e l'FCM diventa la scelta migliore. La PCM-R invece è troppo lenta anche per n = 2.

Metriche	FCM-PF	PCM	KM	FCM
Silhouette (Sil)	0.5641	-0.3541	0.3844	0.5559
Silhouette con Fuzzy (SilF)	0.6239	0.0473	–	0.6904
Xie-Beni (XB)	0.2101	374000.1748	0.2785	0.1445

Table 7: Risultati delle metriche per PF, PCM, KM e FCM.

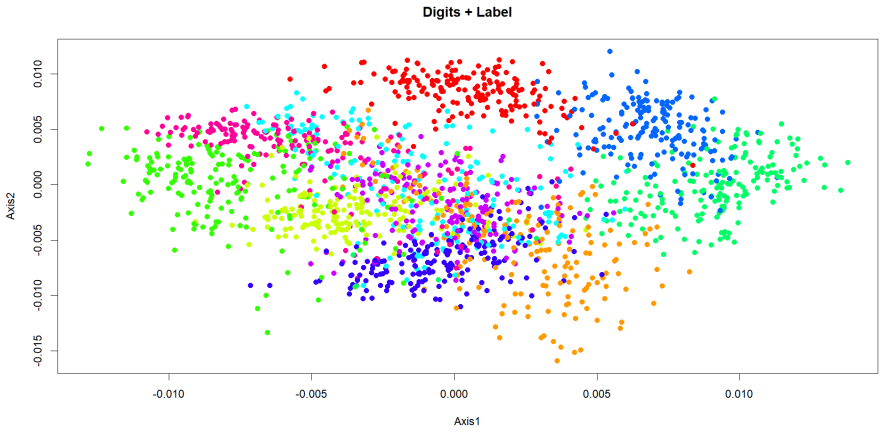


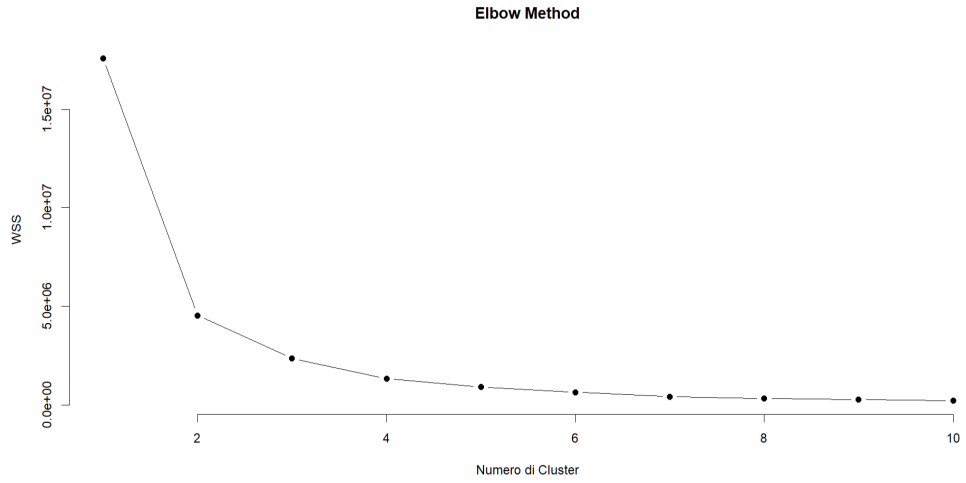
Figure 4: Plot delle Digits con n = 2 e label prese dal dataset

Varianza Spiegata	PC1 + PC2
FPCA	0.6883
PCA	0.2851

Table 8: FPCA VS PCA

4.3 Wine Dataset

In questo caso è stato effettuato anche uno studio sull'Elbow Plot per capire il numero di cluster ottimale.



Una buona scelta per il numero di clustering è 6, a partire da questo valore, la Within-Cluster Sum of Squares (WSS), non diminuisce più in modo drastico.

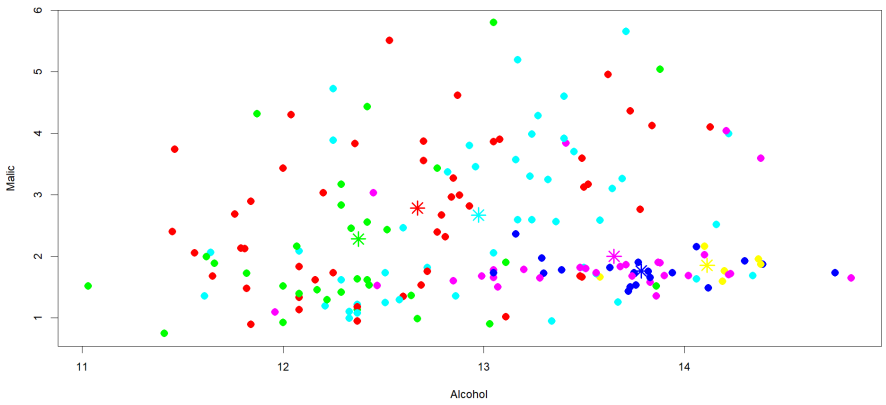


Figure 5: Esempio Plot Clustering su Wine Dataset (FCM)

I risultati, senza FPCA, per gli algoritmi KM, FCM e FCM-PF sono:

Metriche	FCM-PF	KM	FCM
Silhouette (Sil)	0.7125	0.5422	0.7117
Silhouette con Fuzzy (SilF)	0.7383	–	0.7876
Xie-Beni (XB)	0.1904	0.2077	0.1393

Table 9: Risultati delle metriche per PF, KM e FCM.

Qui l'algoritmo FCM sembrerebbe la scelta più adatta, ed è anche l'unico Case Study per cui la FPCA (n = 2, varianza spiegata = 99.8%, equivalente alla PCA classica in questo caso) provoca variazioni significative sugli indici.

Metriche	FCM-PF	PCM	KM	FCM
Silhouette (Sil)	0.6740	0.0247	0.4811	0.6698
Silhouette con Fuzzy (SilF)	0.7130	0.5718	–	0.7718
Xie-Beni (XB)	0.1114	467.8961	0.1643	0.0875

Table 10: Risultati delle metriche per PF, PCM, KM e FCM.

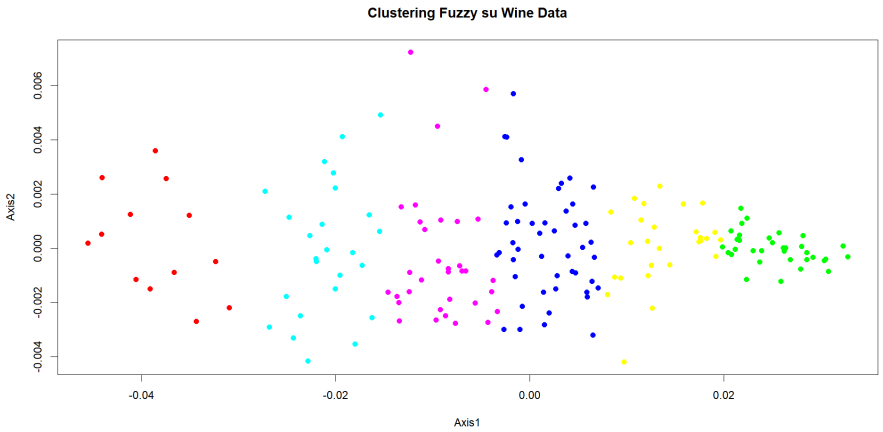
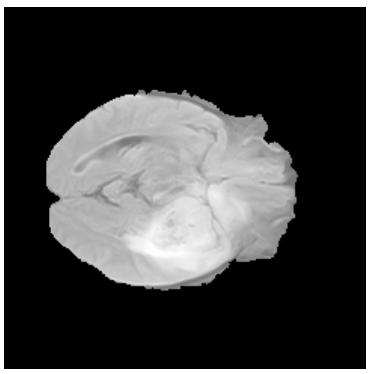


Figure 6: Clustering su Wine Dataset (FPCA), n = 2, k = 6

5 Brain Tumor Segmentation

Il rilevamento automatico dei tumori all'interno di immagini mediche ha un ruolo fondamentale nella diagnosi. Le risonanze magnetiche (RM) sono una delle scelte più popolari per identificare tessuti anomali. Segmentare le RM aiuta a suddividere il tessuto in più regioni, in base a caratteristiche come intensità, colore e texture. Un possibile approccio è il clustering sulle immagini, che raggruppa pixel simili confrontando la distanza di ciascun data point con i diversi centri dei cluster.

I diversi tessuti cerebrali, tra cui il liquido cerebrospinale, la materia bianca e la materia grigia, non sono ben distinguibili nelle RM, di conseguenza, i bordi tra le regioni si confondono tra loro, come nella seguente sezione:



Gli algoritmi di fuzzy clustering assegnano ad ogni data point diversi gradi di appartenenza a ciascun cluster. Di conseguenza, sono comunemente utilizzati per la segmentazione di tumori, permettendo di gestire le incertezz riguardanti la sovrapposizione dei cluster, che rappresentano i tessuti nelle immagini RM. Il caso di studio in oggetto è la segmentazione di RM per individuare tumori cerebrali utilizzando l'algoritmo FCM.

Qui sfrutteremo l'estensione di Gustafson-Kessel (GK), che utilizza la metrica di distanza di Mahalanobis. Si tratta di un metodo per determinare la similarità di uno spazio

campionario incognito rispetto ad uno noto. Rispetto alla distanza euclidea tiene conto delle correlazioni all'interno dell'insieme dei dati. La variante GK calcola inizialmente le matrici di covarianza F_i per ciascun centro dei cluster.

$$\mathbf{F}_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m (\mathbf{x}_j - \mathbf{c}_i)(\mathbf{x}_j - \mathbf{c}_i)^\top}{S_i}, \quad 1 \le i \le C$$

$$S_i = \sum_{j=1}^{N} \mu_{ij}^m$$

Successivamente, calcola la distanza di Mahalanobis da ciascun punto dati a ciascun cluster utilizzando le matrici di covarianza.

$$D_{ij} = \sqrt{(\mathbf{x}_j - \mathbf{c}_i)^\top \left[\det(\mathbf{F}_i)^{1/N} \mathbf{F}_i^{-1}\right] (\mathbf{x}_j - \mathbf{c}_i)}, \quad 1 \le i \le C, \quad 1 \le j \le N$$

Quindi la covarianza del cluster è ponderata in base ai valori di appartenenza al cluster. Questo metodo è utile per rilevare cluster non sferici [8].

Infatti nel caso in cui la distribuzione è ad esempio iperellissoidale, la probabilità di quel punto di appartenere all'insieme dipende non solamente dalla distanza dal centro di massa, ma anche dalla direzione. Sulle direzioni in cui l'iperellissoide ha un asse più corto, il punto deve esser più vicino per esser considerato appartenente all'insieme, mentre sulle direzioni in cui l'asse è più lungo, il punto può trovarsi anche a distanze maggiori [2].

5.1 Brain Tumor Segmentation Using FCM Clustering

Il processo di segmentazione include i seguenti passaggi:

1. Preparare i vettori delle caratteristiche dai dati di addestramento e di test.
2. Raggruppare i dati di addestramento utilizzando la metrica di distanza specificata.
3. Identificare il cluster che rappresenta il tumore.
4. Utilizzare i centri dei cluster identificati per individuare i tumori nei dati di test.

Questo esempio utilizza il dataset BraTS, che contiene volumi 4D. Ogni volume 4D ha dimensioni 240×240×152×4, dove le prime tre corrispondono all'altezza, larghezza e profondità dell'immagine volumetrica 3D, mentre la quarta rappresenta diverse modalità di scansione. Noi useremo solo i dati della prima modalità di scansione. Si scaricano due volumi, uno per il training e uno per il test set, insieme alle etichette di test.

```
imageDir = fullfile(tempdir,"BraTS");
filename = matlab.internal.examples.downloadSupportFile(...
    "vision","data/sampleBraTSTestSetValid.tar.gz");
untar(filename,imageDir);

trainDataFileName = fullfile(imageDir,...
    "sampleBraTSTestSetValid","imagesTest","BraTS447.mat");
testDataFileName = fullfile(imageDir,...
    "sampleBraTSTestSetValid","imagesTest","BraTS463.mat");
testLabelFileName = fullfile(imageDir,...
    "sampleBraTSTestSetValid","labelsTest","BraTS463.mat");
```

Per prima cosa, si rimodella la prima modalità di scansione dai dati di addestramento, trasformandola da una matrice di dimensioni 240×240×152×1 in una matrice 57.600×152, in cui ciascuna colonna rappresenta una sezione verticale del dataset volumetrico 3D.

```
orgTrainData = load(trainDataFileName);
[r,c,n,~] = size(orgTrainData.cropVol);
trainingData = reshape(orgTrainData.cropVol(:,:,:,1),[r*c n]);
```

Successivamente, si creano i vettori delle caratteristiche dai dati di training per catturare i pattern spaziali nei dati e si utilizza una finestra mobile su ciascun pixel nel training, appiattendo la sotto-immagine acquisita nella finestra in un vettore riga di caratteristiche. Dimensioni grandi della finestra possono portare a sovrapporre diversi pattern nella stessa sottoimmagine.

In questo esempio, si utilizza una finestra 3×3 che genera un vettore di caratteristiche con nove elementi per ciascun data point. Poi si crea una matrice delle caratteristiche con nove colonne utilizzando la funzione "createMovingWindowFeatures". Ogni riga della matrice corrisponde ad un data point.

```
kDim = [3 3];
trainFeatures = createMovingWindowFeatures(trainingData,kDim);
```

Si fa la stessa cosa per i dati di test.

```
orgTestData = load(testDataFileName);
testData = reshape(orgTestData.cropVol(:,:,:,1),[r*c n]);
testFeatures = createMovingWindowFeatures(testData,kDim);
```

Ora bisogna caricare i dati etichettati per le immagini di test, che indicano quali pixel nei dati di test rappresentano tumori.

```
orgLabel = load(testLabelFileName);
refLabel = orgLabel.cropLabel;
```

Quindi identifichiamo gli indici dei pixel etichettati come tumore per ciascuna immagine e impostiamo un flag se un'immagine contiene pixel tumorali.

```
refTumor = cell(1,n); % Tumor pixel ids
refHasTumor = false(1,n);
for id = 1:n
    refTumor{id} = find(refLabel(:,:,id)==1);
    refHasTumor(id) = ~isempty(refTumor{id});
end
```

Ora possiamo passare alla segmentazione dell'immagine. Per prima cosa si stabiliscono le opzioni per l'algoritmo FCM.

```
numClusters = 15;
options = fcmOptions;
options.NumClusters = numClusters;
options.MaxNumIteration = 10;
options.MinImprovement = 1e-3; % Criterio di arresto su miglioramento
options.DistanceMetric = "mahalanobis";
```

Si esegue quindi il clustering dei dati di training con la funzione fcm. Per ottenere risultati coerenti, si inizializza anche il generatore di numeri casuali prima della call a fcm.

```
rng("default")
mnCenters = fcm(trainFeatures, options);
```

Calcola la distanza di ciascun data point nei dati di test da ciascun centro dei cluster calcolati.

```
mnDist = findDistance(mnCenters,testFeatures);
```

Per ciascun data point, trova l'indice del centro di cluster più vicino.

```
[~,mnLabel] = min(mnDist',[],2);
```

Rimodella l'array degli indici dei centri di cluster per farlo corrispondere alle dimensioni originali dell'immagine. mnLabel conterrà l'etichetta del cluster per ciascun pixel dell'immagine nei dati di test.

```
mnLabel = reshape(mnLabel,n,r*c)';
mnLabel = reshape(mnLabel,[r c n]);
```

Ora identifichiamo i pixel tumorali in ciascuna immagine di test utilizzando la funzione di supporto "segmentTumor". Per utilizzare questa funzione, si specifica l'indice del cluster che rappresenta il tumore. Possiamo identificare manualmente il cluster tumorale nei dati di addestramento (in questo caso 12). La funzione "segmentTumor" esegue le seguenti operazioni:

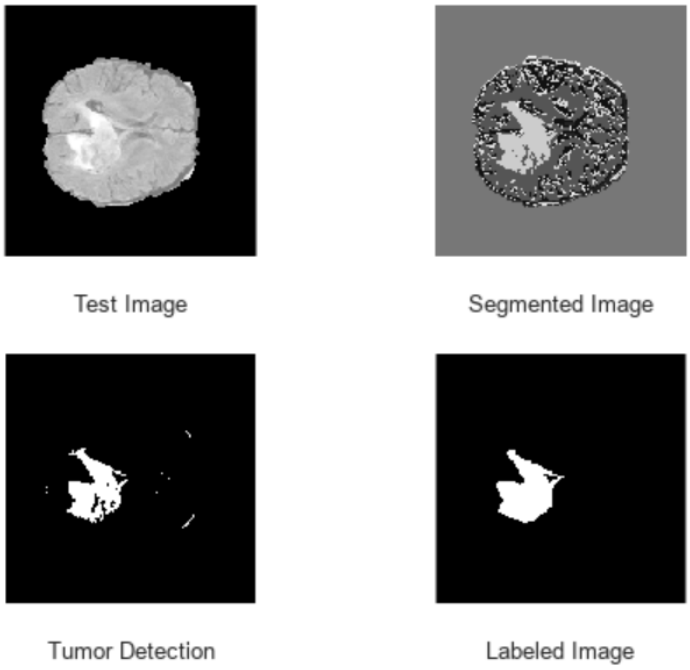
- Etichetta i pixel tumorali nell'immagine.
- Calcola il numero di pixel falsi positivi.
- Restituisce un valore logico che indica se l'immagine contiene un tumore.

```
mnTumor = zeros(r,c,n);
mnHasTumor = zeros(1,n);
mnNumFalsePos = zeros(1,n);
tumorCluster = 12;

for id=1:n
    [mnHasTumor(id),mnNumFalsePos(id),mnTumor(:,:,id)] = ...
        segmentTumor(mnLabel(:,:,id),refTumor{id},tumorCluster);
end
```

mnLabel(:, :, id) è una matrice bidimensionale che rappresenta la segmentazione dell'immagine id-esima del dataset dopo l'applicazione del clustering Fuzzy C-Means (FCM).

Poi possiamo fare il plot dei risultati con le quattro immagini per il caso di test:



5.2 Confronto con Distanza Euclidea (FCM Classico)

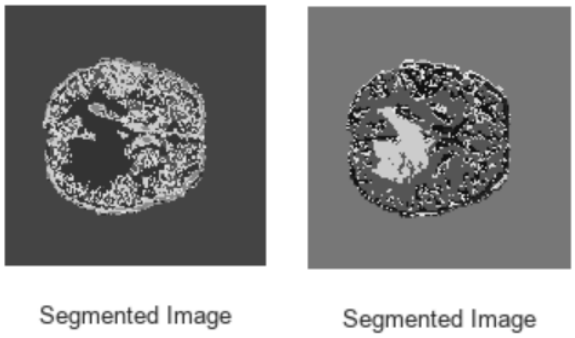
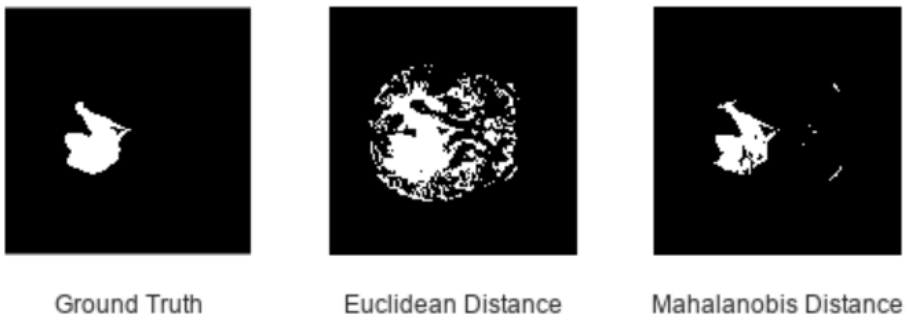
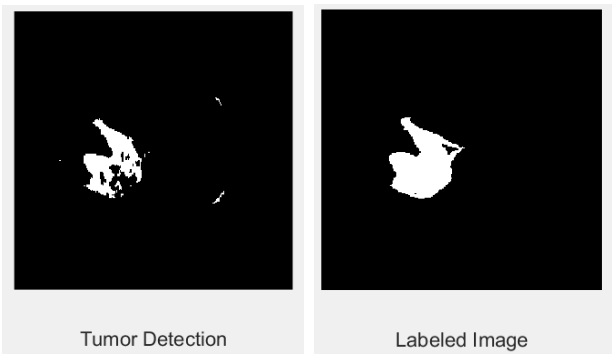


Figure 7: Immagine Segmentata con FCM Classico a Sinistra, con Variante GK a Destra



	True Pos	True Neg	False Pos	False Neg
Euclidean Distance	80	26	46	0
Mahalanobis Distance	79	36	36	1

5.3 Confronto con K-Means



Il K-Means, nonostante trovi meno falsi positivi, non riesce bene a cogliere le forme dell'intero tumore, proprio per via dell'esistenza di aree sfumate. In realtà, siccome ci sono piccoli falsi positivi nelle soluzioni trovate dall'algoritmo FCM, una cosa che si può fare è trovare le componenti connesse, misurare la loro area, e rimuovere le componenti che non superano una piccola area di pixel. Risultati:

True Pos	True Neg	False Pos	False Neg
79	61	11	1

5.4 Funzioni di Supporto

Per catturare i pattern spaziali nei dati, è necessario creare vettori che rappresentino il contesto locale di ciascun pixel. A tal fine, si utilizza una finestra mobile 3×3 per analizzare le proprietà locali. Ciò permette di rappresentare ogni pixel attraverso i valori dei suoi vicini, migliorando l'analisi dei pattern e riducendo il rumore.

```
function y = createMovingWindowFeatures(in,dim)
%% Create feature vectors using a moving window.

rStep = floor(dim(1)/2);
cStep = floor(dim(2)/2);

x1 = [zeros(size(in,1),rStep) in zeros(size(in,1),rStep)];
x = [zeros(cStep,size(x1,2));x1;zeros(cStep,size(x1,2))];

[row,col] = size(x);
yCol = prod(dim);
y = zeros((row-2*rStep)*(col-2*cStep), yCol);
ct = 0;
for rId = rStep+1:row-rStep
    for cId = cStep+1:col-cStep
        ct = ct + 1;
        y(ct,:) = reshape(x(rId-rStep:rId+rStep,cId-cStep:cId+cStep),1,[]);
    end
end
end
```

Questa funzione estrae caratteristiche locali da un'immagine 2D (matrice) trasformandola in una rappresentazione numerica più ricca. Il processo avviene in diversi passaggi: Definizione del passo della finestra mobile:

```
rStep = floor(dim(1)/2);
cStep = floor(dim(2)/2);
```

Viene calcolato il numero di pixel da considerare su ciascun lato della finestra. Per una finestra 3×3, il valore sarà floor(3/2) = 1, ovvero il raggio di analisi attorno al pixel centrale.

Padding dell'immagine:

```
x1 = [zeros(size(in,1),rStep) in zeros(size(in,1),rStep)];
x = [zeros(cStep,size(x1,2));x1;zeros(cStep,size(x1,2))];
```

L'immagine originale viene circondata da zeri per gestire i bordi senza perdita di dati. Il padding assicura che ogni pixel, compresi quelli al bordo, abbia un'area completa da analizzare.

Preparazione della matrice di output:

```
[row,col] = size(x);
yCol = prod(dim);
y = zeros((row-2*rStep)*(col-2*cStep), yCol);
```

Si calcola il numero di righe e colonne della matrice estesa. La variabile yCol rappresenta il numero di elementi nel vettore di caratteristiche per ogni pixel ($3 \times 3 = 9$). La matrice y verrà popolata con le caratteristiche estratte.

Scansione dell'immagine con la finestra mobile:

```
ct = 0;
for rId = rStep+1:row-rStep
    for cId = cStep+1:col-cStep
        ct = ct + 1;
        y(ct,:) = reshape(x(rId-rStep:rId+rStep,cId-cStep:cId+cStep),1,[]);
    end
end
```

Un doppio ciclo for scorre pixel per pixel. Per ogni posizione (rId, cId), viene estratta una finestra 3×3 attorno al pixel centrale. La finestra viene appianata (reshape(...)) in un vettore di 9 elementi e aggiunta alla matrice y.

La funzione restituisce una matrice in cui ogni riga rappresenta un pixel dell'immagine. Ogni colonna contiene i valori della finestra 3×3 attorno a quel pixel. Questa trasformazione è essenziale per applicare metodi di clustering e classificazione, in quanto fornisce un insieme di caratteristiche che descrivono meglio la struttura locale dell'immagine.

Vogliamo associare ciascun pixel al cluster più vicino. Per questo creiamo la funzione findDistance.

```
function dist = findDistance(centers,data)
%% Calculate feature distance from cluster center.

dist = zeros(size(centers, 1), size(data, 1));
for k = 1:size(centers, 1)
    dist(k, :) = sqrt(sum(((data-ones(size(data, 1), 1)*centers(k, :)).^2), 2));
end
end
```

Definizione della Matrice delle Distanze:

```
dist = zeros(size(centers, 1), size(data, 1));
```

"centers" è una matrice di dimensioni (numClusters × numFeatures), dove ogni riga rappresenta un centroide di un cluster trovato con FCM. "data" è una matrice di dimensioni (numPixels × numFeatures), dove ogni riga rappresenta un pixel caratterizzato dal vettore di caratteristiche estratto con la finestra mobile. "dist" è una matrice di dimensioni (numClusters × numPixels), in cui ogni elemento dist(k, i) memorizzerà la distanza del pixel i dal centroide del cluster k.

Calcolo della Distanza Euclidea:

```
for k = 1:size(centers, 1)
    dist(k, :) = sqrt(sum(((data-ones(size(data, 1), 1)*centers(k, :)).^2), 2));
end
```

Il ciclo for scorre tutti i centroidi k = 1:numClusters e calcola la distanza tra ogni centroide centers(k, :) e tutti i data point usando la formula della distanza euclidea.

Dopo aver calcolato le distanze, si assegna ogni pixel al cluster con la distanza minima. Nel nostro codice, mostrato nella sezione precedente:

$$[\tilde{}, \mathbf{mnLabel}] = \mathbf{min(mnDist', [\,], 2);}$$

Trova l'indice del cluster con la distanza minima per ciascun pixel. mnLabel contiene la mappa delle etichette dei cluster per tutti i pixel dell'immagine.

Infine per segmentare l'immagine:

```
function [hasTumor, numFalsePos, tumorLabel] = segmentTumor(testLabel, refPositiveIds, clusterId)
% Calculate detection results using the test and reference data.

tumorIds = testLabel == clusterId;  % Identifica i pixel assegnati al cluster del tumore
segmentedImage = testLabel;
segmentedImage(tumorIds) = 1;       % Imposta i pixel tumorali a 1
segmentedImage(~tumorIds) = 0;      % Imposta gli altri pixel a 0
tumorIdsECIds = find(tumorIds == 1); % Ottiene gli indici dei pixel segmentati come tumore
hasTumor = ~isempty(tumorIdsECIds);  % Verifica se almeno un pixel è stato classificato come tumore
numFalsePos = length(find(setdiff(tumorIdsECIds, refPositiveIds))); % Conta i falsi positivi
tumorLabel = segmentedImage;         % Restituisce l'immagine binaria segmentata
end
```

Il codice originario è presente in:

https://it.mathworks.com/help/fuzzy/brain-tumor-segmentation-using-fuzzy-c-means-clustering.html

Il sito non presenta tuttavia il confronto col K-Means, nè prova a ridurre il numero di falsi negativi restituiti dall'FCM.

References

[1] Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt. Advances in intelligent data analysis v. In *5th International Symposium on Intelligent Data Analysis, IDA 2003, Berlin, Germany, August 28-30, 2003, Proceedings*, pages 262–263, Berlin, Germany, 2003. Springer. Conference proceedings.

[2] Wikipedia contributors. Distanza di mahalanobis. `https://it.wikipedia.org/wiki/Distanza_di_Mahalanobis`, 2024. Accessed: 2024-02-11.

[3] Copernicus Publications. Supplementary material for "egu sphere 2023-1896". `https://egusphere.copernicus.org/preprints/2023/egusphere-2023-1896/egusphere-2023-1896-supplement.pdf`, 2023. Accessed: 2024-12-11.

[4] I. Couso, C. Borgelt, E. Hullermeier, and R. Kruse. Fuzzy sets in data analysis: From statistical foundations to machine learning. *IEEE Computational Intelligence Magazine*, 14(1):31–44, Feb. 2019.

[5] T. Cundari, C. Sârbu, and H. Pop. Robust fuzzy principal component analysis (fpca): A comparative study concerning interaction of carbon–hydrogen bonds with molybdenum-oxo bonds. *Journal of Chemical Information and Computer Sciences*, 42:1363–1369, 2002.

[6] Luiz Carlos de Barros, Rodolfo C. Bassanezi, and Weldon A. Lodwick. *A First Course in Fuzzy Logic, Fuzzy Dynamical Systems, and Biomathematics: Theory and Applications*. Springer, Cham, 2017.

[7] Raghu Krishnapuram and James M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[8] MathWorks. Fuzzy clustering, 2024. Accessed: 2024-02-11.

[9] Mohammad Rawashdeh and Anca L. Ralescu. Fuzzy cluster validity with generalized silhouettes. In *Midwest Artificial Intelligence and Cognitive Science Conference*, 2012.

[10] H. Timm, C. Borgelt, C. Döring, and R. Kruse. An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147(1):3–16, 2004.

[11] Vicenç Torra, Yasuo Narukawa, and Sadaaki Miyamoto. Modeling decisions for artificial intelligence. In *Second International Conference, MDAI 2005, Tsukuba, Japan, July 2005, Proceedings*, volume LNAI 3558, pages 159–160. Springer, Berlin, Germany, 2005.

[12] Wikipedia contributors. Covarianza — wikipedia. https://it.wikipedia.org/wiki/Covarianza_(probabilit%C3%A0), 2024. Accessed: 2024-12-11.

[13] Wikipedia contributors. Indicatriceinsieme — wikipedia. https://it.wikipedia.org/wiki/Funzione_indicatrice, 2024. Accessed: 2024-12-11.

[14] Wikipedia contributors. Membershipfunct — wikipedia. https://en.wikipedia.org/wiki/Membership_function_(mathematics), 2024. Accessed: 2024-12-11.

[15] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.